



FileOpen WebPublisher3

Copyright 1998-2007 FileOpen Systems Inc. All rights reserved.

This manual, and the software described in it, is furnished under license and may be used only in accordance with the terms of that license. The content of this manual is for informational use only, is subject to change without notice, and should not be construed as a commitment on the part of FileOpen Systems Inc. which assumes no responsibility or liability for errors or inaccuracies that may appear in this document.

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written consent of the copyright owner, FileOpen Systems Inc., P.O. Box 28, Red Hook, NY 125714. The copyrighted software that accompanies this manual is licensed to the end user for use only in strict accordance with the End User License Agreement, which the Licensee should read carefully before commencing use of the software.

Adobe™ and Acrobat™ are registered trademarks of Adobe Systems Inc.

Macintosh is registered trademark of Apple Computer, Inc.

Microsoft is a registered trademark, and MS-DOS and Windows are trademarks of Microsoft Corporation.

All other trademarks are owned by their respective claimants.

STRUCTURE OF THE SYSTEM	5
Architectural Overview.....	6
OPERATION OF THE SYSTEM.....	7
Encryption.....	7
Decryption.....	7
Identification of Users.....	8
IMPLEMENTATION OF THE SYSTEM.....	9
Running the Encryptor	9
The License File.....	10
The Encryptor Initialization Files	10
Encryptor Syntax	10
Parameter File Elements	11
Setting up the Client.....	14
The Trace File	15
Setting up a PermissionServer Process	16
Communication Protocol Requests.....	17
Communication Protocol Elements	19
Generic Request Elements	19
Generic Response Elements.....	23
Communications Protocol Syntax.....	27
The Setting Request	27
Structure of the Setting Request	28
Structure of the Setting Response	28
Modifying the Acrobat Alert box:	29
Modifying the Username/Password (UNP) Dialog:	30
Document Access Control	31
The DocPerm Request	31
Behavior of SendAnyway:.....	32
Positive Response to DocPerm:	32
The Document Timeout Request	33
Negative Response to DocPerm:	34
Replacing the Authentication Data via XAuthent:	36
Hashing the User's Password:	37
Displaying a Conditional Dialog:	37
DocPerm Values	39
Basic DocPerm Values	39
Extended DocPerm Values	41
Rules for Extended DocPerms:.....	41
The "jumpMode" and "Message" Actions:	42
DocPrint Notification.....	43
DocClosed Notification	43
DocSaved Notification.....	45
Document Print Control.....	46

The PrintPerm Request	47
Document Text Copy Control.....	49
Approaches to Copy Control	52
Granting Offline Permission	53
Details of Offline Permission Modes.....	53
Creating and Updating the Offline Permission List.....	55
Mechanism for Granting Static Offline Permission.....	55
Structure of the Offline Permission File	56
Offline Expiration	57
The Action= Statement	58
Mechanism for Granting Dynamic Offline Permission	60
Inverted Offline Permissions	61
Local Offline Permission Value Reporting.....	61
Offline Failure Notification	62
Location of Offline Permission File	63
Dynamic Forms Manipulation	64
Use of the System in an Intranet Environment	65
THE SAMPLE SCRIPTS	66
TROUBLESHOOTING.....	66
Appendix 1: Document and User Identification Constants	67
APPENDIX 2: DETAILS OF ONLINE AND OFFLINE MODES.....	69
APPENDIX 3: SETTING UP THE ACTIVEX AND JAVA INSTALLERS	70
Logic of the Install Procedure	70
Entries in the Auto-Install Page	73
The TargetUrl parameter:	73
The CookieUrl parameter:	74
The 128-bit Installer, binary:	75
APPENDIX 4: USER AUTHENTICATION MODES.....	77
Static or Dynamic Encryption:.....	77
Online or Offline Access:	77
Relative Usefulness of Authentication Methods:	78
Cookie Authentication:	78
Username/Password Authentication:.....	78
Machine Authentication:.....	78
Machine Identifiers	79
ERRORS DEFINED BY PDFENCRYPTOR2	80
ERRORS DEFINED BY PDFENCRYPTOR3	81
ACROBAT LANGUAGE CODES	82
DOCUMENT PERMS	83

Structure of the system

FileOpen WebPublisher3 consists of a pair of applications, one to encrypt PDF documents (the **Encryptor**) and another to decrypt and display those documents (the **Client**). A third element (the **PermissionServer**) is also required to manage interaction with the client via the communication protocol, however this functionality is considered to be outside of the system and is provided only in example form.

These applications employ a common metadata scheme (the document data) and the client exposes a simple communications protocol. The two applications form the core of a publishing system. A complete system requires the addition of server-side software to define the publisher's business logic (e.g. to identify a document, then to determine whether a given user should be allowed to open that document) and to communicate this business logic to the client via the PermissionServer.

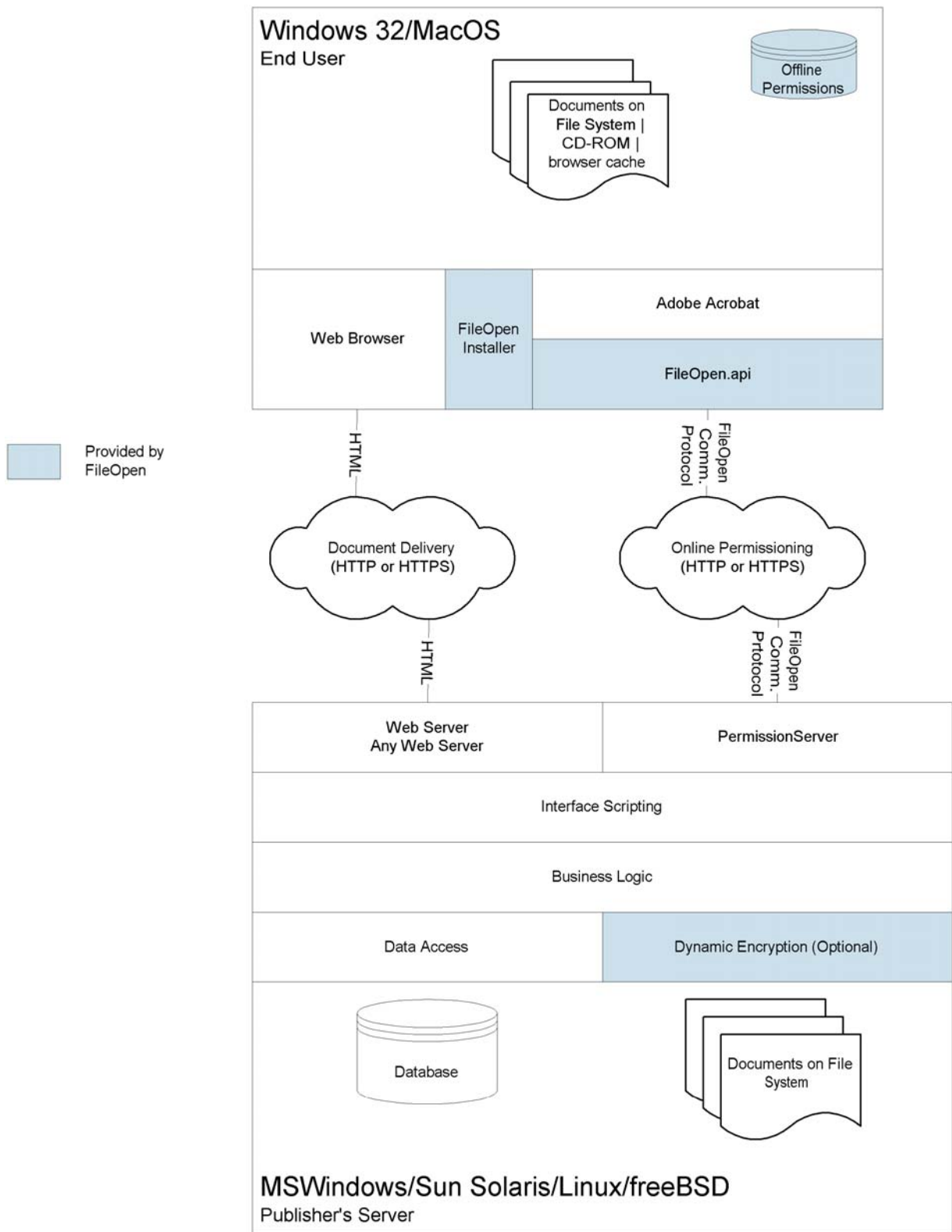
Three classes of service are provided:

- **Online Access:** The default state, in which all documents are opened from a remote server, typically within a web browser, and the decryption key for each document is retrieved from the publisher's server via the client-server communication described below.
- **Online/Offline Hybrid Access:** In this case the document is stored on and opened from the user's local machine, but the permission to open that document is obtained via the client-server communication mechanism.
- **Offline Access:** In this mode the user may open documents from local storage (hard disk or CD-ROM) without contacting the publisher's server. The ability to open documents in Offline mode is contingent upon the user having a valid Offline Permissions file, generated by the publisher's server via a mechanism described below and maintained on the local machine by the FileOpen client.

These modes are discussed in detail in an Appendix 2.

The general Adobe Acrobat security model is described in Appendix 5.

Architectural Overview



Operation of the System

The system works in two steps, Encryption and Decryption. The Encryption step may be performed either in advance of document distribution or in real time, i.e. on a server as the document is being distributed to the end user. The decryption step is always done as the document is being opened by the end user, but the Decryption Key (which is symmetric, i.e. is the same as the Encryption Key) may be delivered either ahead of time (offline mode) or in real time (online mode).

Encryption

WebPublisher encrypts documents in accordance with the PDF specification, details of which are available from Adobe Systems¹. PDF files are encrypted by the system using RC4 with either a 40 or 128-bit key.

During the encryption step, each document is assigned an encryption key and a set of metadata, described below. All metadata other than the Encryption Key is stored in the document. The Encryption Key is used to encrypt the document, and then is discarded. By design, all metadata elements are defined by the publisher, subject to the above limitations.

The encryption of documents and insertion of metadata is performed via a commandline operation, described below.

Decryption

During the decryption step, the Client application retrieves from the document a block of configuration metadata to determine the location of the publisher's PermissionServer and the parameters to use in contacting that server.

Once the location of the publisher's PermissionServer has been obtained, the Client initiates a connection to that server via http or https (specified by the publisher) and passes a structured request for permission to open the document. This request, expressed in the form of the Communications Protocol, must then be parsed and evaluated by the PermissionServer, which should return either a positive answer (the Decryption Key for the document and a set of permissions) or a negative one (refusal to open the document and a reason).

¹ <http://partners.adobe.com/asn/tech/pdf/specifications.jsp>

A third case, in which the user is granted permission to open an arbitrary set of documents for a limited amount of time without further contact with the server (Offline Mode), is also supported and is discussed [below](#).

Identification of Users

The opening of documents via WebPublisher3 is governed by the publisher's own logic applied to a question which can be reduced logically to the form:

Action((DocumentID +UserID))?

Where **Action** is one of the Communication Protocol Requests, **DocumentID** is defined by the publisher, then retrieved from the document and passed to the publisher's server and **UserID** may be defined in a number of ways. Among the possible definitions of a User are:

- login to the publisher's server, represented by a session cookie on the local machine retrieved by the Client
- login to the publisher's server, represented by a username/password stored in a cookie on the local machine retrieved by the Client
- username/password pair obtained by the Client via a dialog
- machineID obtained from the local machine by the Client
- some other data, e.g. IP address, obtained by the PermissionServer from the header accompanying the client request

One of these options must be specified during the encryption of a document, via the mechanism described in the next section. A complete list of elements that must be defined during encryption is given [below](#).

Further information on the types of authentication mechanisms is provided in an Appendix.

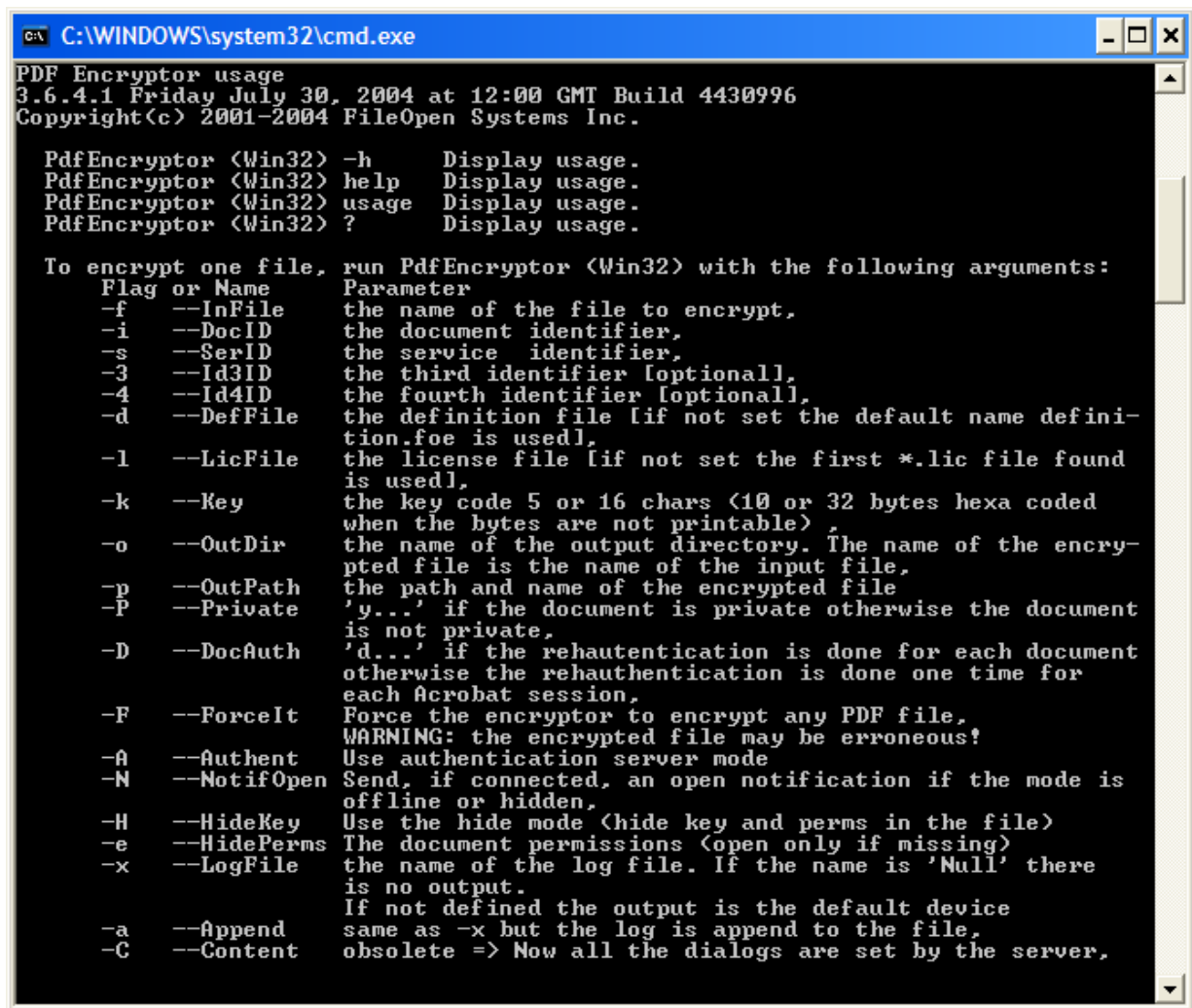
Implementation of the System

The following steps are required to develop a publishing system with FileOpen WebPublisher3:

Running the Encryptor

The Encryptor is available on multiple computer platforms (MSWindows, Sun Solaris, Linux, FreeBSD, HP-UX), and is operated as a commandline-driven executable and as a library. Syntax is the same for all platforms, and is given by running the program with the arguments “-h” or “help” or “usage” or “?”. Errors reported by the encryptor are described in an appendix to this document.

The usage syntax for the PDFEncryptor is shown below. Syntax for and operation of the PDFEncryptor is the same whether files are being encrypted offline on a desktop system or online on a server. It is irrelevant, from the perspective of the client software, when and where the file was encrypted.

A screenshot of a Windows command prompt window. The title bar reads "C:\WINDOWS\system32\cmd.exe". The window contains the following text:

```
PDF Encryptor usage
3.6.4.1 Friday July 30, 2004 at 12:00 GMT Build 4430996
Copyright(c) 2001-2004 FileOpen Systems Inc.

PdfEncryptor <Win32> -h      Display usage.
PdfEncryptor <Win32> help    Display usage.
PdfEncryptor <Win32> usage   Display usage.
PdfEncryptor <Win32> ?      Display usage.

To encrypt one file, run PdfEncryptor <Win32> with the following arguments:
Flag or Name      Parameter
-f --InFile       the name of the file to encrypt,
-i --DocID        the document identifier,
-s --SerID        the service identifier,
-3 --Id3ID        the third identifier [optional],
-4 --Id4ID        the fourth identifier [optional],
-d --DefFile      the definition file [if not set the default name defini-
tion.foe is used],
-l --LicFile      the license file [if not set the first *.lic file found
is used],
-k --Key          the key code 5 or 16 chars <10 or 32 bytes hexa coded
when the bytes are not printable>
-o --OutDir       the name of the output directory. The name of the encry-
pted file is the name of the input file,
-p --OutPath      the path and name of the encrypted file
-P --Private      'y...' if the document is private otherwise the document
is not private,
-D --DocAuth      'd...' if the reauthentication is done for each document
otherwise the reauthentication is done one time for
each Acrobat session,
-F --ForceIt     Force the encryptor to encrypt any PDF file,
WARNING: the encrypted file may be erroneous!
-A --Authent     Use authentication server mode
-N --NotifOpen   Send, if connected, an open notification if the mode is
offline or hidden,
-H --HideKey     Use the hide mode <hide key and perms in the file>
-e --HidePerms   The document permissions <open only if missing>
-x --LogFile      the name of the log file. If the name is 'Null' there
is no output.
-a --Append      If not defined the output is the default device
same as -x but the log is append to the file,
-C --Content     obsolete => Now all the dialogs are set by the server,
```

The License File

A license file with the suffix .lic must be present in the directory from which the PDFEncryptor program is run. A file demouser.lic is provided along with the evaluation components of the system. For licensed use, this file should be deleted or renamed and the publisher's own license file inserted into the directory containing the PDFEncryptor.

The Encryptor Initialization Files

The PDFEncryptor application obtains metadata from an initialization file, known as the **Parameter File**. This file is read by the encryptor to obtain the base parameters to use when processing the document specified in the commandline. The Parameter File is mandatory. An annotated version of this file is provided in the evaluation distribution with the name *definition.foe*.

Some data in the Parameter File may be overridden by data provided to the encryptor via the commandline syntax.

Encryptor Syntax

The PDFEncryptor encrypts a single document at a time. For every document, the encryptor must be given all the required parameters either as commandline arguments or entries the Parameter File.

Some parameters are not supported in the commandline, so must be specified in the Parameter File. As a rule, these are parameters (e.g. the URL to the publisher's PermissionServer, the type of user identification, etc.) that will not change from document to document. Where a parameter is supported in both the commandline and the Parameter File, the presence of that parameter in the commandline will override any value given in the Parameter File.

An example of the Parameter File is given below.

```
header=fOpenWebPub
DocumentID=TestFile1
ServiceID=C-ABC
Ident3ID="K-1-Ident 3"
Ident4ID=Identifier4ID
CryptoKey=abcde
OfflinePerms=yes
Authent=Cookies
CommUse=session
ErrorMode=SendAnyway
LogUrl=http://www.fileopen.com:80/fowp/index_asp.htm
UserUrl= http://www.fileopen.com:80
DocPerm=/bin/fowp.asp
FilePerm=/bin/fowp.asp
CookUrl=www.fileopen.com
CookPath=/
CookSessTag=sess
ServerName=WebPublisherDemonstrator
Agent="Acrobat Reader FileOpen WebPublisher Plug-in"
DataFolder=WebPublisherDemo
```

Parameter File Elements

An annotated version of the example file follows:

(text in blue is reserved and must be used as presented, text in grey is variable).

header=fOpenWebPub

Usage: Required in Parameter File

Format: Must be as shown.

Function: Flags the Parameter File for the encryptor.

Syntax: none

DocumentID=TestFile1

Usage: Required in Parameter File unless overridden in Commandline

Format: Any 63 character string

Function: Identifies the document

Syntax: none

ServiceID=C-ABC

Usage: Required in Parameter File unless overridden in Commandline

Format: Any 63 character string

Function: Further identifies the document

Syntax: none

Ident3ID="K-1-Ident 3"

Usage: Optional

Format: Any 63 character string

Function: Further identifies the document

Syntax: none

Ident4ID=Identifier4ID

Usage: Optional

Format: Any 63 character string

Function: Further identifies the document

Syntax: none

CryptoKey=abcde

Usage: Required in Parameter File unless overridden in Commandline

Format: 5 or 16 byte string²

Function: Used to encrypt

Syntax: none

OfflinePerms=yes

Usage: Optional

Format: A Constant

² A key must have exactly 40 or 128 bits. These may be represented as:

- A string of 5 or 16 characters ASCII characters if they are printable
- A string of 10 or 32 hexadecimal characters (0..9,A..F)

Entries in the .foe file may be quoted and contain spaces: the syntax "**CryptoKey=04257**" is valid, as is "**CryptoKey= 04257**". However, if the key itself contains spaces, e.g. " abc ", the syntax "**CryptoKey= abc** " is erroneous (the string would be parsed as "abc") and instead must be written **CryptoKey=" abc "** or "**CryptoKey= abc** "

Encryption keys specified in the command line must be explicitly defined, as the commandline interpreter doesn't know the type of the values being given. A commandline argument -k 04257 will be parsed as the number 4257 and generate an error. The correct syntax is -k "04257". To avoid any trouble with the command line parameters it is best to always quote the strings.

Function: Specifies whether Offline Permissions should be private or public³
Syntax: "Yes" or "y" or "Y" or "Private" for private, "No" or "N" or "n" or "Public" for Public

Authent=Cookies

Usage: Required in Parameter File
Format: A Constant
Function: Specifies what data client should retrieve from local machine context
Syntax: Choose from:
 Cookies (or "c")
 Machine (or "m")
 PasswordBox (or "p")

CommUse=session

Usage: Required if Authent=Cookies
Format: A Constant
Function: Further specifies what data client should retrieve from local machine context
Syntax: Choose from:
 Session (or "s")
 Useraname (or "u")

ErrorMode=SendAnyway⁴

Usage: Optional
Format: A Constant
Function: If Authent=c, this specifies that the client should make a DocPerm request even if the cookie is not found or the UNP data has not been provided.
Syntax: SendAnyway or NULL

LogUrl=http://www.fileopen.com/fowp/index_asp.htm, or
http://www.fileopen.com:80/fowp/index_asp.htm, or
https://www.fileopen.com:443/fowp/index_asp.htm

Usage: Required
Format: A valid⁵ URI string
Function: The client will re-direct a user to this URL if:
 - authent=c and ErrorMode is not SendAnyway
 - the requested cookie is not found
Syntax: The port argument is optional. If not present, port 80 is used; use 443 for https.

UserUrl= <http://www.fileopen.com:80>

Usage: Required
Format: A valid URI string
Function: The base URL for DocPerm and FilePerm requests
Syntax: The port argument is optional. If not present, port 80 is used; use :443 for https.
Notes: See LogUrl for port syntax.

DocPerm=/bin/fowp.asp

Usage: Required
Format: A valid URI substring.

³ Private Offline Permissions are stored in a form that requires the user to enter a username and password to open documents. This is useful for shared environments, like libraries, in which multiple users may be downloading documents to the same machine. Public Offline Permissions are specific to a machine and a login, but not to a particular username/password pair.

⁴ The SendAnyway behavior will vary based on the value of Authent=, see *Behavior of SendAnyway* below.

⁵ See <http://www.ietf.org/rfc/rfc2396.txt>

Function: Appended to the UserUrl to make the URI used for **DocPerm** requests
Syntax: No arguments supported.
Notes: **PrintPerm** requests are sent to the script referenced for **DocPerm** requests.

FilePerm=/bin/fowp.asp

Usage: Required
Format: A valid URI substring.
Function: Appended to the UserUrl to make the URI used for **DocPerm** requests
Syntax: No arguments supported.
Notes: The same script can be referenced for **DocPerm** and **FilePerm**.

CookUrl=www.fileopen.com

Usage: Required if Authent=c
Format: A valid URI substring.
Function: The domain from which the cookie will be set⁶
Syntax: No arguments supported.

CookPath=/

Usage: Required if Authent=c
Format: A valid URI substring.
Function: Specifies the path, if any, to the cookie
Syntax: No arguments supported.

CookSessTag=sess

Usage: Required if Authent=c
Format: A valid URI substring.
Function: Specifies key within the cookie for which a value should be obtained.
Syntax: No arguments supported.

ServerName=WebPublisherDemonstrator

Usage: Required
Format: Any 63 character string
Function: Specifies the name of the server that will be sent in the header of the request
Syntax: No arguments supported.

Agent="Acrobat Reader FileOpen WebPublisher Plug-in"

Usage: Required
Format: Any 63 character string
Function: Specifies the name of the agent in the header of the request sent to the server.
Syntax: No arguments supported.

DataFolder=WebPublisherDemo

Usage: Required
Format: Any 63 character string
Function: Specifies the name of the file folder to be created for offline permission files.
Syntax: No arguments supported.

⁶ It is permissible to set a cookie for all servers in a particular domain by prepending a dot to the server name. If CookUrl=.fileopen.com the client will find cookies from foo.fileopen.com and bar.fileopen.com.

Setting up the Client

The client software is implemented as a plug-in, FileOpen.api, to the Adobe Acrobat viewer. The plug-in will load in either the full version of Acrobat or in the Adobe Reader; versions 4.0 or later are supported⁷. The client is currently available for Microsoft Windows, where it will load in Windows95/OSR2 or any later version, and for Apple Macintosh, where it will load in OSX 10.4 or later, and for Linux.

The Windows distribution of the client contains an optional companion plug-in, fowp[*l*]kbd.api. This plug-in implements the client control over screen capture.⁸

There is no requirement that the client and companion be installed programmatically; however both must be located in the proper place on the user's drive in order to be loaded by Acrobat. By default, this location is:

Windows:

C:\Program Files\Adobe\Acrobat 4.0\5.0\6.0\7.0\Acrobat\Reader\plug-ins

Macintosh:

Varies by version and user preference. Plugin should be installed by installer or via Apple+I:Plugins:Add

Linux:

Can be placed in any location, see installer Readme.

The Client and companion may be placed in this location manually. Alternately, an installer is provided as ***FileOpenInstaller.exe*** and ***FileOpenInstaller.msi*** on Windows and ***FileOpenInstaller.sit*** or ***FileOpenInstaller.dmg*** on Macintosh. The Windows installer has been signed with a MS Authenticode signature. It will perform the following steps:

1. Detect the presence or absence of Adobe Acrobat. If Acrobat is not found on the local machine a message is presented to the user saying that Acrobat must be installed.

⁷ The plug-in will also load in all known "intermediate" versions of Acrobat, e.g. Acrobat Business Tools. However, some documents require version 5 or later, see 128-bit Installer, below.

⁸ There are two versions of the keyboard companion plugin, for different versions of Windows. On Windows9x and NT4 the fowp3kbd.api is used, else fowp4kbd.api. If the fowp[*l*]kbd.api plug-in is present, the client will notify the server in the DocPerm request by adding a statement Fowpkbd=Yes. If the companion plug-in is not present the main plug-In will operate normally but screen capture cannot be prevented.

2. Detect the Acrobat preference to load only Adobe plug-ins. If this preference is set, the installer will request permission to change it.
3. Install the plug-in to all local copies of Adobe Acrobat.
4. Present a message stating that Installation is Complete.

The installer is also provided in a form that may be used to install the client from within the Microsoft Internet Explorer browser (or any other browser that supports ActiveX). This version of the installer is distributed as FileOpen.cab and is explained in more detail in Appendix 3.

The Trace File

The WebPublisher3 distribution contains two versions of the FileOpen.api client:

FileOpen.api – the release client (freely distributable)

FileOpenTrace.api – the debug client (provided only for testing, not to be distributed externally)

Each time the FileOpenTrace.api is run it will generate a text file with the name fowptrace.txt; you can find this file by searching the disk of the local machine⁹. The tracefile contains a record of the functions performed by the client since the plug-in was loaded along with a listing of parameter data retrieved from each file and a recording of all communication between the client and the PermissionServer. Each time the plug-in is loaded, i.e. each time Acrobat is launched, the tracefile will be overwritten.

Note that in order to run the FileOpenTrace.api you must remove FileOpen.api – both plug-ins cannot load at the same time in the same copy of Acrobat.

If you encounter any issues with documents are not opening you should read this file after each event to determine the source of the error. All interaction between the client and server is logged in the file, though some data is masked¹⁰.

Searching on the name of a Communication Protocol keyword (DocPerm, PrintPerm, etc.) will locate that interaction and may help to identify the source of the error.

⁹ The file is typically written to c:\documents and settings\<login>\application data\fileopen; note that if this folder is hidden a search won't find the file either until the folder is made visible (Show Hidden Files and Folders).

¹⁰ For example, decryption keys are not displayed in the file. To get the complete dataset for an client/server interaction one must have access to both the tracefile and the log of the PermissionServer.

Setting up a PermissionServer Process

The evaluation distribution contains the code used in the Sample Server application, written in Perl and ASP. There is no requirement that a PermissionServer be written in any particular language. Provided that it conforms to the syntax of the **Communications Protocol**, the PermissionServer may be written in any language run on any platform. Pre-built server components and complete “turnkey” PermissionServer systems are available from third party vendors affiliated with FileOpen Systems; for more information on third party PermissionServers see www.fileopen.com.

The Communications Protocol consists of a set of structured requests and responses. The system invokes two general forms of communication:

Queries: are requests that require an answer, given in the form:

Request= followed by the kind of request¹¹.

Notifications: are requests that do not require an answer, given in the form:

Info= followed by the kind of notification¹².

The current set of requests is given below and described in more detail in the following section:

Every Query or Notification contains the following parameter elements:

- **Basic class** (sent with all server exchanges): Mode, Stamp; ServiceID, DocumentID, Ident3ID, Ident4ID, Machine, Disk, Uuid, UserName, UserPass, UNPData, Session, User, USR
- **Version class:** ProdVer, EncrVer
- **Acrobat & Document class:** AcroVersion, AcroProduct, AcroCanEdit, InBrowser, DocIsLocal, DocPathUrl
- **Extended class:** Build, Language, FowpKbd, OfflineFail

In addition, some requests or notifications may contain other specialized pairs:

- **Previous state identifiers:** PrevMach, PrevDisk
- **Printing:** Count, PageRanges, Printer
- **Acrobat printing:** AcroPrt
- **Copying:** CopySelected, CopyPageFrom, CopiedTotal, CopyRequired

¹¹ Currently the requests are: *DocPerm, FilePerm, KeepOpen, Setting, SaveOffLinePerms, PrintPerm, AcroPrint, MaxWordsCopied*

¹² Currently the notifications are: *DocOpened, DocClosed, DocPrinted, AcroPrint, WordsCopied, PageViewed, DialogClosed*

- **Dialog closed:** Reason
- **Offline Permission State:** OfflineOpens, OfflineExpire, OfflinePrints

Communication Protocol Requests

Request=Setting

Usage: Once per Acrobat session

Function: Allows the PermissionServer to specify strings and modes for the client

Response: Optional

Present in version: Client Private Build 0500 or later.

Request=DocPerm

Usage: Each time a document is opened, if offline permissions for that document are not present

Function: Made by the client to request the decryption key and permission settings for the document being opened.

Response: Required

Present in version: All versions

Request=PrintPerm

Usage: Each time the user attempts to print a document, if print permission was granted using PrintControl¹³

Function: Made by the client to request the permission to print the document.

Response: Required

Present in version: All Versions

Request=FilePerm

Usage: Once per open attempt on a PermissionPDF

Function: Made by the client to request Offline Permission File.

Response: Required

Present in version: All Versions

Request= SaveOffLinePerms

Usage: Each time the user attempts to save a local copy of a document, if the SavePerm notification was requested and plugin has loaded full Acrobat, not Reader.

Function: Made by the client to request Dynamic Offline Permissions.

Response: Required

Present in version: Client Private Build 0500 or later

Request=KeepOpen

Usage: Sent when a document timeout has been reached, if the AskServer parameter was set during encryption.

¹³ With **PrintControl** the user's request is sent to the PermissionServer, via the PrintPerm request, before any printing is initiated. With **PrintMonitoring** the print job takes place at the user's discretion and notification is sent to the PermissionServer upon completion.

Function: Permits the server to reset the document timeout and, optionally, to present a message to the user.

Response: Required

Present in version: Client Private Build 0500 or later

Request=MaxWordsCopied

Usage: Sent when the user has attempted to copy a block of text exceeding the value given by the server in the MaxWordsCopy parameter in response to either the DocPerm request or a previous instance of this request, when CopyControl was requested.

Function: Permits the server to grant or deny the copy action, and optionally to reset the value of MaxWordsCopy

Response: Required

Present in version: Client Private Build 0650 or later.

Info=DocOpened

Usage: Sent when the document is opened, if **NotifOpen** flag was placed into document during encryption.

Function: Notifies PermissionServer that document has been opened, if the user is connected. If no internet connection is available, the notification is not sent.

Response: None

Present in version: Client Private Build 0550 or later.

Info=DocPrinted

Usage: Sent when document is printed, if notification was requested.

Function: Notifies PermissionServer that print job is complete.

Response: None

Present in version: Client Private Build 0500 or later.

Info=DocClosed

Usage: Sent when the document is closed, if notification was requested.

Function: Notifies PermissionServer that document has been closed.

Response: None

Present in version: All Versions

Info=WordsCopied

Usage: Sent when the user has copied text from the document, if CopyControl was request.

Function: Notifies PermissionServer that text has been copied.

Response: None

Present in version: Client Private Build 0650 or later

Info=AcroPrint

Usage: Sent when the Acrobat print dialog is used and a print done notification is required

Function: Notifies PermissionServer that printing is complete.

Response: None

Present in version: Client Private Build 0650 or later

Info= DialogClosed¹⁴

¹⁴ Note that the request for this notification must be inserted into the document during encryption. It cannot be requested by the PermissionServer because the client behavior may occur prior to the client contacting that server.

Usage: Sent when the UNP dialog is closed, in the form:
Info=DialogClosed&...&Reason=DialogCancelled | RetriesCount | Unknown
Function: Notifies PermissionServer that the UNP dialog was dismissed.
Response: None
Present in version: Client Private Build 0650 or later

Info=PagesViewed¹⁵

Usage: Sent when the NotifyPages=true command has been received and when a new page is displayed.
Function: Notifies PermissionServer that the user has changed the page view.
Response: None
Present in version: Client Private Build 720 or later

Communication Protocol Elements

The Communication Protocol Requests will contain some combination of the following protocol elements, and the PermissionServer's response must be in one of the forms given.

Generic Request Elements

The following Elements may appear in any Request; additional Request Elements, if any, are specific to particular Requests and described below in the context of those requests.

Mode=

Purpose: Informs the server of the reason the request is being made
Source: Generated by the client from the local machine context.
Format: One of a set of pre-defined strings¹⁶

Stamp=

Purpose: A timestamp generated by the client to prevent caching of requests and responses
Source: Generated by the client from the local machine context.
Format: UTC integer, e.g. Stamp=1074022019

ServiceID=

Purpose: Identifies the document and/or the group of which the document is a member
Source: Generated by the publisher and inserted into the document during encryption
Format: Any 63 character string

DocumentID=

Purpose: Identifies the document

¹⁵ This notification will be in the form Info=PagesViewed&...&Pages=<from>,<to>&... where <from> is the previous page and <to> the new page, i.e. a switch from page 1 to page 2, from 12 to 20, etc.

¹⁶ The Mode= string values are:

Mode=und	Undefined, irrelevant
Mode=CNR	Offline Permission Classic No Record (no record in the file or no file)
Mode=CEC	Offline Permission Classic Expired/Consumed
Mode=ICx	Any Auxiliary Permissions

Source: Generated by the publisher and inserted into the document during encryption
Format: Any 63 character string

Machine=

Purpose: Identifies the local machine context¹⁷

Source: Generated by the client

Format: A character string

PrevMach=

Purpose: Indicates that the MachineID obtained from the current machine context is not the same as the one found previously on that machine. In this case the current MachineID is sent as Machine= and the previous one in this element.¹⁸

Source: The local machine context

Format: A character string

Uuid=

Purpose: Identifies the local machine context¹⁹

Source: Generated by the client

Format: A character string

UuidIsNew=

Purpose: Indicates whether the UUID has been generated for this request, either because the machine is being accessed for the first time or because an existing UUID was not found or was corrupted.

Source: The local machine context

Format: A constant: **Yes** (UUID is new). If the UUID is not new no value for UuidIsNew is sent²⁰.

Disk=

Purpose: Identifies the local machine context

Source: Generated by the client

Format: A character string

PrevDisk=

Purpose: Indicates that the DiskID obtained from the current machine context is not the same as the one found previously on that machine. In this case the current DiskID is sent as Disk= and the previous one in this element.

Source: The local machine context

Format: A character string

VolName=

Purpose: Identifies the name, if any, of the volume from which the file is being read.

Source: Generated by the client

Format: A character string

¹⁷ The Machine= identifier is derived algorithmically from the local machine context. Details of the method are available upon request.

¹⁸ Values for PrevDisk and PrevMach are sent only in the first DocPerm & FilePerm requests In any given session, and are also sent with the Setting and in KeepOpen requests.

¹⁹ The UUID is generated according to DCE 1.1, see <http://www.opengroup.org/onlinepubs/9629399/apdx.htm>

²⁰ If a new UUID is generated the previous one is not retrievable; there is only one UUID, either the existing one or a new one, hence no value for PreviousUUID.

VolType=

Purpose: Identifies the type of volume from which the file is being read (fixed disk, CD, network drive, etc.).

Source: Generated by the client

Format: A character string

VolSN=

Purpose: The serial number of the hard disk.

Source: The local machine context

Format: A character string

FSName=

Purpose: The type of FileSystem (e.g. NTFS, HFS).

Source: The local machine context

Format: A character string

OSType=

Purpose: The type of Operating System

Source: The local machine context

Format: A character string

OSName=

Purpose: The product name of the Operating System

Source: The local machine context

Format: A character string

OSVersion=

Purpose: The version of the Operating System

Source: The local machine context

Format: A character string

OSBuild=

Purpose: The build number of the Operating System

Source: The local machine context

Format: A character string

Session=

Purpose: The value of the session cookie retrieved from the local machine, if any

Source: The cookie placed by the PermissionServer

Format: A character string

ProdVer=

Purpose: The public version number of the client making the request

Source: Internal to the client

Format: A character string in the form x.x.x.x²¹, e.g. ProdVer=1.5.0.0

Build=

Purpose: The Private Build Description of the client making the request

Source: Internal to the client

²¹ This identifier is rarely used; we refer to Build= to define the client version.

Format: A character string, e.g. Build=0613

Language=

Purpose: The language identifier for the user's Acrobat viewer

Source: The Acrobat viewer application

Format: A three-letter character string, e.g. Language=ENU²²

FowpKbd=

Purpose: Specifies whether the companion plug-in is loaded (Windows only)

Source: Internal to the client

Format: A constant; FowpKbd=Yes (loaded) or FowpKbd=No (not loaded)

AcroVersion=

Purpose: Gives the version identifier for the user's Acrobat viewer

Source: The Acrobat viewer application

Format: A two digit number, e.g. AcroVersion=6.0

AcroProduct=

Purpose: Describes the type of Acrobat viewer from which the request is being made

Source: The Acrobat viewer application

Format: An Adobe-defined string, e.g. AcroProduct=Exchange-Pro

AcroCanEdit=

Purpose: Further identifies the Acrobat viewer from which the request is being made

Source: The Acrobat viewer application

Format: An Adobe-defined string. AcroCanEdit=Yes indicates full Acrobat; AcroCanEdit=No indicates Reader

InBrowser=

Purpose: Indicates whether the document is being opened from within a Web Browser

Source: The local machine context

Format: A constant: Yes (in the browser), No (not in browser), Unk (state is unknown)²³.

DocIsLocal=

Purpose: Indicates whether the document is being opened from the local file system of the user's machine

Source: The local machine context

Format: A constant: Yes (file is local), No (file is not local), Unk (state is unknown).

DocPathUrl=

Purpose: provides the path from which the PDF is being opened

Source: The local machine context

Format: Either a URL or, if the file is being opened from local disk, a path.

USR=

Purpose: Specifies the Unique Serial Root used for encryption of the document

Source: The document

Format: A five-character string.

²² The list of Acrobat language codes is provided in an appendix to this document.

²³ Note that this value is obtainable only when the document has been opened, so its value in the DocPerm request will always be "unk".

Generic Response Elements

The following Elements may appear in any Response; additional Response Elements, if any, are specific to particular Requests and are described below in the context of those requests.

RetVal=

Usage: Required unless otherwise indicated²⁴

Purpose: Indicates whether server's response is positive, negative or some extended value

Format: Dependent on the request, though generally 0 for No, 1 for Yes, 2 for Update²⁵.

Code=

Purpose: Is the decryption key for the document

Source: The publisher's server logic.

Usage: In response to DocPerm request, if RetVal=1.

Format: A string as defined under **CryptoKey**, [above](#).

Perms=

Purpose: The Permission State for the session.

Source: The publisher's server logic.

Usage: In response to DocPerm request, if RetVal=1.

Format: A string or bitmask as defined under DocPerm Values, [below](#).

Pext=

Purpose: The Extended Permission State for the session.

Source: The publisher's server logic.

Usage: In response to DocPerm request, if RetVal=1.

Format: A string or bitmask as defined under DocPerm Values, [below](#).

ServId=

Purpose: The ServiceID for the document in question

Source: The **ServiceID** value from the input request

Format: A string; should be returned as received.

Doculd=

Purpose: The DocumentID for the document in question

Source: The **DocumentID** value from the input request

Format: A string; should be returned as received.

Error=

Purpose: Specifies an error condition, with explanation following.

Source: The publisher's server logic.

Usage: In response to DocPerm or PrintPerm request, if RetVal=0.

Format: A string of arbitrary length.

Action: The string following Error= will be displayed to the user.

Message=

Purpose: Specifies a message to be shown to the user, with the text of the message following.

Source: The publisher's server logic.

²⁴ The string "RetVal" is treated as case-sensitive by the client, so must be sent as "RetVal"; any other formulation ("retVal", "RetVal", etc.) will generate an error.

²⁵ The client will accept either the numeric or text formulation of the response, that is::

"RetVal=error" is the same as "RetVal=0"

"RetVal=answer" is the same as "RetVal=1"

"RetVal=update" is the same as "RetVal=2"

Usage: In response to DocPerm, PrintPerm or MaxWordsCopied request, if RetVal=1.

Format: A string of arbitrary length.

Action: The string following Message= will be displayed to the user.

Dialog=

Purpose: Specifies a dialog to be shown to the user, with the text of the message, and containing an OK and Cancel button.

Source: The publisher's server logic.

Usage: In response to DocPerm, PrintPerm or MaxWordsCopied request.

Format: A string of arbitrary length.

Action: The string following Dialog= will be displayed to the user; see *Displaying a Conditional Dialog* below for a description of this action.

Redirect=

Purpose: Specifies an error condition, with explanation following²⁶.

Source: The publisher's server logic.

Usage: In response to DocPerm request.

Format: A valid URL.

Action: If RetVal=1, the client will resend the request to the URL specified.

LogUrl=

Purpose: Specifies a URL to be opened in the default browser.

Source: The publisher's server logic.

Usage: As part of the Dialog= response.

Format: A valid URL.

Action: Launches the default browser and opens the specified URL; operation depends on RetVal value, See *Displaying a Conditional Dialog*, below.

Availability: Client private build 0660 or higher.

AcroPrntDlg=

Purpose: Instructs the client to allow printing under **PrintMonitoring** rather than **PrintControl**.

Source: The publisher's server logic.

Usage: In response to DocPerm request.

Format: Reserved words (true or false)

Action: If "true" client will use PrintMonitoring; if "false" (or absent) client will use PrintControl.

Note that this statement will be ignored if the value of **Perms=** does not permit printing.

Pnot=

Purpose: Instructs the client to send **PrintNotification** to the PermissionServer

Source: The publisher's server logic.

Usage: In response to DocPerm request or PrintPerm request

Format: Reserved words

Action: Will send **Info=DocPrinted** notification as requested.

BlockScreenShot=

Purpose: Instructs the client to restrict copying the screen via PrintScrn and Alt+PrintScrn

Source: The publisher's server logic.

Usage: In response to DocPerm request.

Format: Reserved words (true or false).

Action: If "true" client will implement screenshot prevention; if "false" (or absent) client will not.

This command also disables the Acrobat Snapshot tool.

Availability: Client private build 0505 or higher.

²⁶ Several types of Redirect are supported, see "The DocPerm Request" below for details.

BlockSearch=

Purpose: Instructs the client to disable the Search option for open files

Source: The publisher's server logic.

Usage: In response to DocPerm request.

Format: Reserved words (true or false).

Action: If "true" client will disable the search button and right-click context menu if "false" (or absent) client will not.

Availability: Client private build 0535 or higher. Note that right-mouse-click functionality is available only in Acrobat/Reader 6 and later.

CopyMax=

Purpose: Instructs the client to limit the number of words that may be copied from the document

Source: The publisher's server logic.

Usage: In response to DocPerm request or MaxWordsCopied request.

Format: Integer.

Action: Specifies the allowed number of words that may be copied from the document.

Availability: Client private build 0650 or higher.

CopyMaxAction=

Purpose: Specifies the behavior of the client when the CopyMax limit has been reached.

Source: The publisher's server logic.

Usage: In response to DocPerm request or MaxWordsCopied request.

Format: One of several reserved words.

Action: If "None" or "0" no action is performed and no additional copying permitted. If "Ask" or "1" the client will make a MaxWordsCopied request when the CopyMax limit is reached. If "Close" or "2" the client will close the document when the CopyMax limit is reached.

Availability: Client private build 0650 or higher.

CopyAdd=

Purpose: Specifies the number of words to add to the CopyMax limit.

Source: The publisher's server logic.

Usage: In response to DocPerm request or MaxWordsCopied request.

Format: Integer.

Action: Increases the number of words that may be copied from the document. Note that if CopyMax and CopyAdd are defined in the same response the CopyMax value used will be CopyMax+CopyAdd.

Availability: Client private build 0650 or higher.

NotifyCopy=

Purpose: Instructs the client to sent notification when text is copied from the document.

Source: The publisher's server logic.

Usage: In response to DocPerm request or MaxWordsCopied request.

Format: Boolean.

Action: If "true" the client will send notification of copy events via the NotifyCopy notification; if "false" (or absent) no notification will be sent and any previously requested notification is cancelled.

Availability: Client private build 0650 or higher.

XAuthent=

Purpose: Instructs the client to use a different authentication method.

Source: The publisher's server logic.

Usage: In response to DocPerm request.

Format: Reason=XAuthent&SwitchTo=<mode>

Action: Instructs the client to use the mode specified after SwitchTo. See *Replacing the Authentication Data via XAuthent* below for more explanation.

Availability: Client private build 0703 or higher.

NotifyPages=

Purpose: Instructs the client to send notification when a new page of the document is displayed.

Source: The publisher's server logic.

Usage: In response to DocPerm request.

Format: Boolean.

Action: If "true" the client will send notification when a new page is displayed, via the Info=PagesViewed notification; if "false" (or absent) no notification will be sent and any previously requested notification is cancelled.

Availability: Client private build 0720 or higher.

Communications Protocol Syntax

When a user attempts to open a document in Online mode, the client will obtain the address of a PermissionServer from the document and initiate a series of structured requests to that server. The order and internal structure nature of these requests depends on the document data and the response of the PermissionServer, but at a minimum the client will make two requests:

Request=Setting - permits the PermissionServer to initialize the client

Response: optional²⁷

Request=DocPerm - request for permission to open document

Response: mandatory

If the PermissionServer grants permission to print, the client will typically make a **PrintPerm** request at least once per print attempt. If the document being opened is a PermissionPDF, used to deliver Offline Permissions, the client will make a **FilePerm** request.

In addition, several notifications are supported. All of the above are described in detail in the following section.

The Setting Request

This request is presented by the client once per server per Acrobat session, in order to permit the PermissionServer to customize the behavior and appearance of the client's user interface. In the absence of a response the client uses the default settings.

The modification of two dialogs is supported:

- The Acrobat Alert displayed when the authentication mode is 'cookies', the cookie type is 'session' or 'unp' and the client is unable to find the cookie on the local machine.²⁸
- The Username/Password dialog box.

In order to ensure that a user opening documents from different publishers will see the settings provided by the appropriate publisher, the values returned in response to the Settings request by the PermissionServer are stored by the client in a hierarchical tree. Publishers may also specify the level at

²⁷ While a response to this request is not required, it is recommended that the PermissionServer respond with some value (e.g. RetVal=0&Error=Unsupported Request) rather than let the client time-out waiting for a response.

²⁸ This assumes that either the encryption flag 'SendAnyway' is not set or the flag is set and the server responded to the DocPerm request with an error message.

which settings are to be stored, permitting a given server to support multiple sets of settings for different documents or collections of documents.

Settings are stored in one of the following levels:

0. 'Default' (the FileOpen default settings),
1. 'Domain'
2. 'Publisher'
3. 'Service'

The server may determine at which level the structure is stored by defining the 'Target' value in the response to the Settings request. Note that when a new target is defined at the same level as a target that has already been defined, the older one is replaced. If the target is not defined it is assumed to be 'Server'²⁹.

When a setting is required, the client will search the tree in ascending order, looking first for a structure at the deepest level ('Service') then searching the higher levels. If no structure is found, the default settings, i.e. those from the 'Default' level, are used³⁰.

Structure of the Setting Request

The Setting request contains the following Protocol Elements:

`Request=Setting&Stamp=&ServiceID=&DocumentID=&Machine=&Uuid=&Session=
&ProdVer=&Build=&Language=&FowpKbd=&AcroVersion=&AcroProduct=&AcroCanEdit=&InBrowse=&DocIsLocal=`

Structure of the Setting Response

The PermissionServer's response should always contain the following:

`RetVal=&SetTarget=&SetScope=`

Where values for **Target** are "UnpDlg" | "AlertLog" (or numeric values 1 | 2) and

Values for **Scope** are "Domain" | "Publisher" | "Service" (or numeric values 1 | 2 | 3)

²⁹ Domain is defined as the name of the domain, not the URL. If the value is 'www.somedomain' the 'www.' is removed.

³⁰ Note that application of the Scope/Publisher/Service is governed by the following rules:

Domain scope: the domain of the PermissionServer URL must be the domain from which the settings response is given.

Publisher scope: the license used to encrypt all documents must match the one used for the document that triggered the settings request.

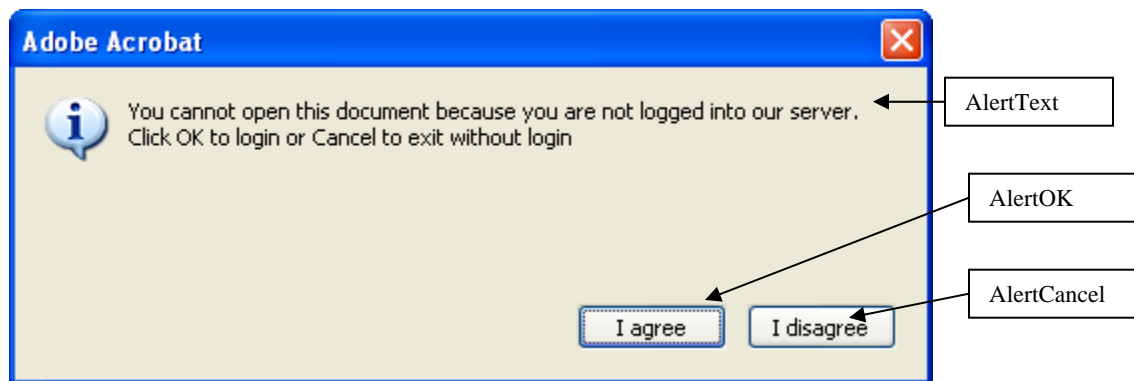
Service scope: the ServiceID applied to all documents must match the one used for the document that triggered the settings request.

Additional Response Elements are specific to the dialog to be modified, as described below. Note that each SetTarget response must be followed by a SetScope response, even if all targets have the same scope. Note also that all elements of the response are required and must be fully formed, i.e. you must include a key/value pair for each element.

An example of this syntax may be obtained by opening a file from the test site at <http://vob.net/fowp> using the trace version of the client, and examining the fowptrace.txt file.

Modifying the Acrobat Alert box:

The Acrobat Alert box is defined as SetTarget=AlertLog. Response Elements for the Alert Box are as follows:



AlertTarget=<target>

Where <target> = "Domain" | "Publisher" | "Service"

AlertText=<text>

Where <text> is the string to display in the Alert³¹.

AlertOk=<text>

Where <text> is the label of the OK button

AlertCancel=<text>

Where <text> is the label of the Cancel button

AlertAct=<action>

Where <action> is the function to execute when the OK button is clicked. In the present version only one action is supported: 'Login'

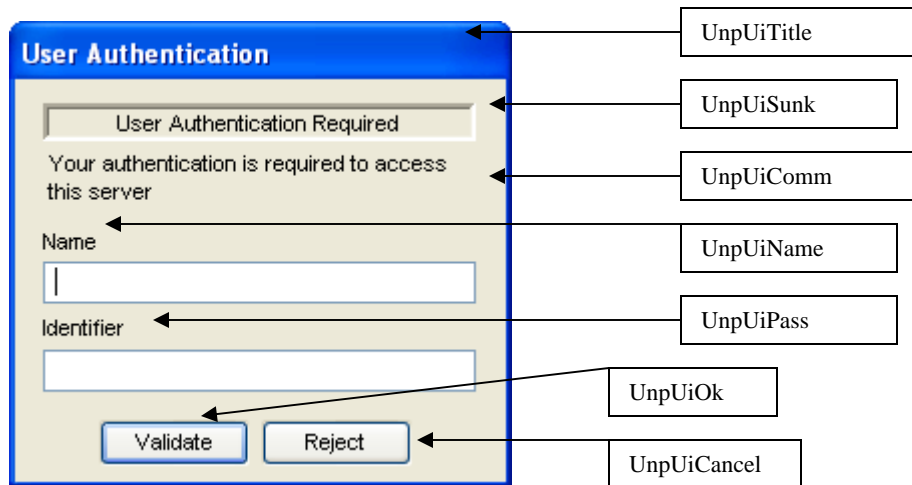
AlertParm=<parms>

Where <parms> is the parameter of the action. In the case of 'Login' the parameter is an Url.

³¹ Alert text is limited to 1023 characters. The text may contain '\n' (carriage return) but must be escaped to support reserved and other non-transferable characters.

Modifying the Username/Password (UNP) Dialog:

The UNP dialog is defined as SetTarget=UnpDlg. Response Elements for the UNP dialog are as follows:



UnpUiTarget =<target>

Where <target> = "Domain" | "Publisher" | "Service"

UnpUiTitle=<title>

Where <title> is the title of the window

UnpUiSunk=<text>

Where <text> is the text displayed in the sunken area. If this value is missing, the title is displayed in this area also.

UnpUiComm=<comm>

Where <comm> is the comment displayed under the sunken area. This text may be presented on multiple lines, if the newline ('\n') character is used.

UnpUiName=<name>

Where <name> is the text displayed above the 'name' input

UnpUiPass=<pass>

Where <pass> is the text displayed above the 'password' input

UnpUiOK=<text>

Where <text> is the label of the OK button

UnpUiCancel=<text>

Where <text> is the label of the Cancel button

Document Access Control

Documents encrypted by the PDFEncryptor may only be opened after the intervention of the client plug-in, which will retrieve the decryption key and permission state either from the PermissionServer (Online access) or the local PermissionFile (Offline Access).

In Online access, the document-open event is passed to the PermissionServer via the DocPerm request.

The DocPerm Request

Unless offline permission has been provided, all attempts to open documents encrypted by WebPublisher will be routed to the PermissionServer server for approval, via the DocPerm request. In the simplest case, the request from the client is of the form:

```
Request=DocPerm&Stamp=&ServiceID=&DocumentID=&Machine=&Uuid=&<USERDATA>=
&ProdVer=&Build=&Language=&FowpKbd=&AcroVersion=&AcroProduct=&AcroCanEdit=&InBrow
ser=&DocIsLocal=
```

Where **<USERDATA>** is determined by the **Authent=** and **CommUse=** statements introduced into the document during the encryption process. If the encryption parameter for user identification was:

Authent=m then **<USERDATA>** is NULL (no value is passed)

Authent=c and *CommUse=s* then **<USERDATA>** is given as **Session=<cookie value>**

Authent=p then **<USERDATA>** is given as

UserName=<username>&UserPass=<password>³²

For a discussion of when to use the different Authentication methods see Appendix 4.

For a discussion of how to MD5 encode the password value see Appendix 5.

Note also that if *Authent=p* the client will send a value for **UNPData=**, where:

UnpData=0 means the source of the UNP data is Unknown

UnpData=1 means that the data was entered into the UNP dialog during the current request

UnpData=2 means the UNP data was cached from an earlier dialog appearance in the same session

UnpData=3 means the UNP data is derived from a hard-coded value (Forced, Missing...)³³

³² Where **<username>** and **<password>** are the values entered by the user into the UNP dialog.

³³ The server may request that the client get new UNP data via the **Reason=AskUnp** directive, described below.

Behavior of SendAnyway:

The document parameter `ErrorMode=SendAnyway` instructs the client to make an initial DocPerm request without having first obtained the requested Authent data. The `SendAnyway` parameter is applicable to Cookie authentication (`Authent=c`) and Username/Password authentication (`Authent=p`). However, the client behavior differs slightly in these two scenarios.

With Cookie authentication, the `SendAnyway` parameter does not affect the client behavior in regard to attempting to retrieve the cookie. Rather, the parameter is interpreted by the client as an instruction to make the DocPerm request in the event that the cookie is not found, rather than performing the default action of raising the login Alert box. Specifying `SendAnyway` with `Authent=c` will result in the client sending DocPerm requests with a value `Session=NULL` when the session cookie cannot be obtained. The server may then accept the request, e.g. by validating the `MachineID` parameter, or reject the request.

With Username/Password authentication, the `SendAnyway` parameter is interpreted by the client as an instruction to send the DocPerm request a first time with dummy values for Username and Password. The server may then either accept the request, e.g. on the basis of an already registered `MachineID`, or response `RetVal=0&Error=AskUnp` to instruct the client to display the UNP dialog and re-submit the DocPerm request with real values.

The `SendAnyway` instruction may also be used to request a new user identification, via the XAuthent mechanism described below.

Positive Response to DocPerm:

The Server's response, if positive, is minimally of the form:

`RetVal=1&ServId=&Doculd=&Perms=&Code=`

Where:

- The `ServId` and `Doculd` values are the same as the `ServId` and `Doculd` values passed in the DocPerm request. Note that if `Ident3ID` and/or `Ident4ID` are specified during encryption and included in the DocPerm request these values must be returned in the `RetVal` response.
- The `Perms` and `Code` values are determined by the `PermissionServer` using its own logic.

Alternately, the Server may respond:

`RetVal=1&Redirect=<redirURL>`

Where:

- RetVal=1 means that the response is positive, as distinct from the case below.
- The **Redirect=** statement will provide a URL to which the request should be sent with the same parameters as those used in the initial request.³⁴

The Document Timeout Request

The Server may also append a statement to the request asking that the client close the document after a specified period. This request should be included in the RetVal response to the DocPerm request, as follows:

&DocuTimeOut=<seconds>**&DocuTOAction=**<action>

Where <seconds> is a number of elapsed seconds, with the count beginning from the time the response is received by the client. When <seconds> has elapsed, the client will perform one of the following actions:

- If the document timeout flag is set to **Close** (the default) the document will be closed.
- If the document timeout flag is set to **AskServer**, the client will make a **KeepOpen** request to the PermissionServer.

The supported values of the document timeout flag are:

- **DocuTOAction=Close** The flag to close the document is set
- **DocuTOAction=Ask** The flag to ask the server the right to keep the document open is set
- **DocuTOAction=Remove** The document timeout is removed

If the initial instruction contains the value DocuTOAction=Ask, then when the DocTimeOut= value has been reached the client will make a **KeepOpen** request, in the form:

Request=KeepOpen**&Stamp=****&ServiceID=****&DocumentID=****&Machine=****&Uid=****<USERDATA>=**
&ProdVer=**&Build=****&Language=****&FowpKbd=****&AcroVersion=****&AcroProduct=****&AcroCanEdit=****&InBrowse=****&DocIsLocal=**

The response must be in the form:

RetVal=0**&KeepOpen=**<boolean>**[&Explain=**<explanation>**]&.....**

Where:

³⁴ This response will cause all further requests during the same Acrobat session to be sent directly to the URL given in the Redirect= response.

KeepOpen=<Boolean> is either a positive or negative response. Acceptable positive responses are “Y” or “Yes” or “y” or “yes” or “1”; acceptable negative responses are “N” or “No” or “n” or “no” or “0”. If the response is negative, the document will be closed. If the response is positive, the document will be kept open for <seconds>, at which point the request will be made again.

Explain=<explanation> (optional) is a string to be presented to the user in a messagebox. This string may be sent following either a positive or a negative response.

The server may also respond with a complete DocuTimeOut response, e.g.:

RetVal=1&ServId=...&DocId=...&...&**DocuTimeOut**=<time>&**DocuTOAction**=<action>

In this case the client will restart the timer with the new TO value time and the new TO action.

Negative Response to DocPerm:

The Server's response, if negative, is of the form:

RetVal=0&**Error**=<ErrorString>

Where:

- **RetVal**=0 is the negative return value
- <ErrorString> is the publisher's response, to be shown to the user in a MessageBox. This may be any printable string of up to 1023 characters³⁵.

Or is of the form:

RetVal=1 & **Login**=<RedURL>

Where:

- **RetVal**=1 is the return value specifying that the client should redirect the user.
- <RedURL>= is the URL to which the user should be redirected. This response is typically used to direct a user to some website from which permission to open the document may be obtained, e.g. an e-Commerce site or login page.

If Username/Password method has been defined for user identification, then the request will be sent with the USERDATA values **UserName**=<username>&**UserPass**=<password>. In this case the server should validate the username/password pair. If the response to the request is permission to open the document, the client will store the username/password pair in memory for the duration of the Acrobat session, unless the server has provided a timeout value.

³⁵ The text may contain '\n' (carriage return) but must be escaped to support reserved and other non-transferable characters.

The server may request that the Username and Password expire after a given period. This is done by appending the following text to the RetVal to the DocPerm request:

`&UserTimeOut=<seconds>`

Where <seconds> is a number of elapsed seconds, with the count beginning from the time the response is received by the client. When <seconds> has elapsed, the client will present the Username/Password dialog box before passing the next DocPerm request to the server.

In the event that the username/password pair is invalid, but the publisher wants to give the user another chance to enter a valid username/password pair, the server should respond to the DocPerm request with:

`RetVal=0&Reason=BadUserPwd`

The client to re-load the UNP dialog; or

`RetVal=0&Reason=BadUserPwd&Error=an error message`

The client will present the error message to the user, then re-load the UNP dialog. Alternately the server may respond with:

`RetVal=0&Error=<Text>`

Where <Text> is any string, which will then be shown to the user in a MessageBox. After that MessageBox has been dismissed the DocPerm request process is complete.

The PermissionServer may also respond to any DocPerm request containing a USERDATA value with:

`RetVal=0[&Error=<Text>]&Reason=AskUnp`

This will instruct the client to present the UNP dialog to the user. This function may be useful in cases where the supplied UNP data is inherited from a previous user/session (i.e. UnpData=2) and the server needs to get the current user's UNP data.

Replacing the Authentication Data via XAuthent:

Client versions later than 0703 support replacement of the Authent= parameter stored in the document. This behavior is intended to support conditional user authentication, for example in the case where a publisher wishes to use cookie authentication when the document is being delivered from the webserver but username/password authentication when the document is accessed from local storage.

The replacement is implemented via a response to the DocPerm request in the form:

RetVal=0&Reason=XAuthent&SwitchTo=<mode>

Where <mode> is an instruction to switch from the original value, i.e. the one placed in the document during encryption, to the value specified after SwitchTo=.

The following are examples of legal syntax for the XAuthent instruction:

- Swap to a 'Machine' authentication (also named an 'Identifiers' authentication)
`RetVal=0&Reason=XAuthent&SwitchTo=Machine`
or
`RetVal=0&Reason=XAuthent&SwitchTo=Idents`
- Swap to a 'Unp dialog box' authentication
`RetVal=0&Reason=XAuthent&SwitchTo=Dialog`
- Swap to a 'Session cookie' authentication
`RetVal=0&Reason=XAuthent&SwitchTo=SesCook&CookDomain=the_cookie_domain
&CookPath=the_cookie_path&CookSess=the_cookie_session_name`
- Swap to a 'Unp cookie' authentication
`RetVal=0&Reason=XAuthent&SwitchTo=UnpCook&CookDomain=the_cookie_domain
&CookPath=the_cookie_path&CookName=the_cookie_user_name&CookPass=the_cookie_password_name`

As always values may be set using a name (or the first letter the name) or using a numeric values (enum). The authentication modes are:

- Machine = 0,
- Idents = 0,
- Dialog = 1,
- SesCook = 2,
- UnpCook = 3

Please note:

- This exchange is only implemented for a response to a 'DocPerm' request
- The document must have the force connection (ErrorMode=SendAnyway) bit set or must use the machine authentication mode otherwise the document encrypted authentication will be applied first then the exchanged one.
- The syntax is VERY restrictive and in the case of any error the client will ignore the exchange request

Hashing the User's Password:

The client may be instructed to send the user's password, when Authent=p, as an MD5 hash rather than in plaintext. This instruction must be inserted into the document (as the client may not have communicated with the server before sending the first UNP data), as follows:

If using the Commandline encryptr add the following to the foe file: "HashPwd = MD5"

If using the C++ Library, there are 3 hash functions:

InitHash() Reset hashing to no hashing.

This is equivalent to *SetHash(kHashingNone)*

SetHash(eHashing eha) with eha one of

kHashingNone: don't hash the password

kHashingMD5B64: MD5 hash + B64 encode

SetHashMD5B64() It is equivalent to *SetHash(kHashingMD5B64)*

Noted: definition of eHashing enums is in Typedef.h

If using the Framework 1.1 and Framework 2.0 libraries:

UnpHashMode = hash;

with 'hash' one of eHashingNone or eHashingMD5B64

Displaying a Conditional Dialog:

The FileOpen.api client supports a conditional response via an OK_CANCEL dialog. This dialog may be raised with either a positive (RetVal=1) or negative (RetVal=0) response. The behavior of the dialog is conditional on the value of the RetVal response and the user's choice of buttons.

The dialog may be requested by returning:

RetVal=<value>&Dialog=<some dialog text>&Action=<>

In the current client only the LogUrl= action is supported. The LogUrl= response should be followed by a valid URL, and the behavior of that dialog will be as follows:

If RetVal=0&Dialog=<some text>&LogUrl=<some URL>³⁶, then:

- if the user clicks OK the client will launch a browser and open the URL
- if the user clicks Cancel, the request will end.

If RetVal=1&Dialog=<some text>&LogUrl=<some URL>³⁷, then:

- if the user clicks OK the client will launch a browser and open the URL
- if the user clicks Cancel, the request will continue normally, i.e. the document will be opened.

³⁶ This mode is useful in cases where the user might be able to obtain access to the document by going to the URL, e.g. RetVal=0&Dialog=Would you like to purchase this document?&LogUrl=http://www.buythedocument.com/

³⁷ This mode is useful in cases where the user is being asked to invoke an optional action, e.g. RetVal=0&Dialog=A newer client is available, would you like to install now?&LogUrl=http://plugin.fileopen.com/

DocPerm Values

The value provided in the **Perms=** statement contained within the RetVal response are used to determine the manner in which the document is opened and the user's ability to interact with that document. This statement may also be modified by the presence of additional arguments in the **Pext=** statement.

Two classes of DocPerm values are supported:

Basic DocPerm Values: map to the PDPerm values defined in the PDF specification; these control the user's ability to open and print the document, copy text and graphics, annotate and change the document.

Extended DocPerm Values: are additional features provided by the FileOpen client; these control the user's ability to save a local copy, to view or print particular pages of the document and to make screen captures via the PrintScr and Alt+PrintScr mechanism.

Basic DocPerm Values

Basic DocPerm values are given in the value following **Perms=** statement in the RetVal response to the DocPerm request. This value may be expressed either as a numeric value or by a reserved word.

If a word is used, only the first letter is tested and is case insensitive, so 'o', 'O', 'op', 'open', and 'ouvre' all have the same permOpen meaning.

To set more than one permission, numeric values must be added and word permissions must be 'listed' using a non alphanumeric separator³⁸.

For example permission to open and print a document may be given as:

5	(1 + 4)
o p	(o[pen]<separator>p[rint])
open print	
ouvre;pr	

The read-only permissions are:

³⁸ This separator cannot be '=' (equal) or ',' (comma), as these are reserved for the protocol. Other characters, e.g. '+' or '*' should be used instead.

permNone: no permission (the document can't be opened!) : 0 or n[one]. If 'none' is used it must not contain spaces ('n o' would generate an error)

permOpen: the document can be opened: 1 or 0x0001 or o[pen]

permPrint: the document can be printed: 4 or 0x0004 or p[rint]

permCopy: parts of the documents can be copied: 16 or 0x0010 or c[opy]

permSave: the document can be saved: 64 or 0x0040 or s[ave]

This value should be provided as an integer. The value of the integer is derived from, and ultimately is translated back into, a binary value in which each bit is interpreted as a flag. The integer values are those used by the Adobe Acrobat application itself, which are:

Bit in perm	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6
Permission to	Open	Reserved	Print	Edit	Copy	EditNotes	Save	Reserved
Signif. Char.	'o'		'p'		'c'		's'	
Decimal value	1		4	8	16	32	64	
Hexa. value	0x01		0x04	0x08	0x10	0x20	0x40	

So if you want:

Bit	0	2	4	6	Value of perm	
Name	Open	Print	Copy	Save	Hex	Decimal
No perm	0	0	0	0	0x00	0
Open	1	0	0	0	0x01	1
Print	0	1	0	0	0x04	4
Copy	0	0	1	0	0x10	16
Save	0	0	0	1	0x40	64
Open & Print	1	1	0	0	0x05	5
Open & Copy	1	0	1	0	0x11	17
Open & Save	1	0	0	1	0x41	65
Print & Save	0	1	0	1	0x44	68
.....						
Open & Print & Save	1	1	0	1	0x45	69
.....						
Open & Print & Copy & Save	1	1	1	1	0x55	85

You may also use the the pdPerms values as defined in the Acrobat SDK, as follows:

Acrobat Internal Value	Code	Letter	Decimal	Hex	Action
#define pdPermOpen	(1 << 0)	'o'	1,	0x01	Open permission
#define pdPermPrint	(1 << 2)	'p'	4,	0x04	Print permission
#define pdPermEdit	(1 << 3)	'e'	8,	0x08	Edit permission
#define pdPermCopy	(1 << 4)	'c'	16,	0x10	Copy permission
#define pdPermEditNote	(1 << 5)	'n'	32,	0x20	Edit note permission
#define pdPermSave	(1 << 6)	's'	64,	0x40	Save permission

In this mode you may build a permission structure either by:

- adding the values: my_perm = pdPermOpen + pdPermCopy
- or
- logically ORing them: my_perm = pdPermOpen | pdPermCopy

NOTE: If the pdPermOpen bit is not set, the document will not be opened.

A complete list of Perms values is provided in the Appendix to this document.

Extended DocPerm Values

Extended DocPerm values are given in the value following **Pext=** statement in the RetVal response to the DocPerm request. The **Pext=** statement is optional. If the statement is absent from the RetVal response, the Extended DocPerms are set to 0, i.e. page-level control is not invoked.

In the current version, the **Pext=** value supports two forms of extended DocPerm permission:

RestrictPageView: instructs the client to control which pages of the document may be viewed by the user.

RestrictPagePrint: instructs the client to control which pages of the document may be printed by the user.

The Extended DocPerm value may be expressed either as a numeric value or by a reserved word.

Bit in perm	Bit 0	Bit 1	Bit 2	Bit 3++
Permission to	RestrictPageView	<i>Reserved</i>	RestrictPagePrint	<i>Reserved</i>
Signif. Char.	none		none	
Decimal value	1		4	
Hexa. value	0x01		0x02	

If the RestrictPageView bit is set, the RetVal string must also contain values specifying which pages are to be blocked and what action to perform when the user attempts to view one of the restricted pages.

Rules for Extended DocPerms:

- If the **RestrictPageView** bit is set, the **PageBlockList** is mandatory and the **PageBlockAction** is optional (if not defined the default value is "std")
- If the **RestrictPagePrint** bit is set, the **PrintBlockList** is mandatory

The **PageBlockList** pair is ' **Vblk**=<list of blocked pages>³⁹

The **PrintBlockList** pair is ' **Pblk**=<list of blocked pages>⁴⁰.

The **PageBlockAction** pair is ' **Vact**=<action>'

³⁹ A list is in the form <list> = **pageRange** ["," <pageRange>]* with:

pageRange = <apage> | <arange>

<apage> = a page number (the first page is page 1)

<arange> = <apage> "-" <apage> (the second page of the range must be greater than or equal to the first page of the range)

⁴⁰ This pair is needed in the DocPerm response to filter the printPageRange requested by the user.

The “jumpMode” and “Message”Actions:

The <action> statement can be used to define the behavior of the Acrobat viewer and optionally to present a message when the user attempts to view a restricted page. This behavior is invoked by setting **action=jumpMode** [", " <message>] with:

- <jumpMode> the jump action to do when the user tries to go to a blocked page
 - **jumpMode**= "s[td]" | "f[irst]" | "l[ast]" | <apage>⁴¹
 - **message**= (Optional). The message to display when the user tries to access a blocked page.
 - **dispMode**= "o[nlyOne]" | "e[achPage]"
 - "o[nlyOne]" (or "0") = display the message only one time (the default value if <dispMode> is missing)
 - "e[achPage]" (or "1") = display the message for each blocked page access

The default standard page jump is “s[td]”. With this setting, when the user attempts to view a blocked page and this action is specified, the client will revert the viewer to the previous visible page or the next visible page of the document, depending on user’s viewing action (i.e. if the user was trying to go to the next page, the viewer will revert to the previous one; if the user was trying to go to a previous page, the viewer will revert to the next one.)

The settings "first", "last" and <apage> may be specified instead of “s[td]”. When the user attempts to view a blocked page and one of these actions is specified, the client will display the first, last or some specific page. Note that with these settings it may be possible to trap the user on the first page unless the user knows a viewable page and types that page number in the status bar (for example, if **jumpMode=f** and the current page is 1 and the second page is blocked, the user’s attempt to click on the next button will always revert the viewer to the first page.)

The message statement will cause the client to display a messagebox containing the text sent by the PermissionServer when the user attempts to access a blocked page. Note that the messagebox will be displayed only one time unless **dispMode=e**, in which case the message will be displayed each time the user attempts to view a blocked page.

⁴¹ Only the first character of the jumpMode statement only is checked

DocPrint Notification

The PermissionServer may request to be notified when a document has been printed by sending the string **Pnot=<printNotification>** to the client. This request may be made either in the RetVal response to the DocPerm request or in the RetVal response to the PrintPerm request. The notification is supported in both requests in order to permit both **PrintControl**, in which printing is contingent upon the PrintPerm request, and **PrintMonitoring**, in which printing is performed without a preceding PrintPerm request⁴².

If the flag is set in the RetVal response to both the DocPerm and the PrintPerm request, the latter will be used to override the former. This flag is optional - if not defined no print notification is done. Only the first character after the "=" is checked.

DocPrint Notification Values:

- "never" = no print notification is done (this is the default value)
- "done" = the notification is sent when printing is completed⁴³
- "eachPage" = the notification is sent for each printed page

The above values may also be represented numerically:

- 0 == "never"
- 1 == "done"
- 2 == "eachPage"

The notification from the client will be in the form:

info=DocPrinted&Stamp=&ServiceID=&DocumentID=&Machine=&Uuid=[&Session=]

Note : The value '&Session=' will be passed only if cookie authentication is being used.

The PermissionServer should not respond to this notification.

DocClosed Notification

The PermissionServer may request to be notified when a particular document has been closed⁴⁴ by sending the string **Notifyclose=<value>** to the client. This request must be made in the RetVal response to the DocPerm request.

⁴² For further discussion of these terms see *Print Control*, below.

⁴³ The current client does not track printing to the output device, so "done" should be understood to mean "when Acrobat reports that the print job is in the print queue."

⁴⁴ Note that when documents are opened in the browser, it is not always obvious to a user when a document has been closed. A user who opens a document in this way and then follows a link, or types in a new address, will see the previous document disappear but will not have "closed" the document (i.e. that document will still be open in Acrobat, and accessible to the user via the browser's back button). A PDF is closed when the Acrobat File>Close

DocClosed Notification Values:

The DocClosed Notification is optional, so must be requested. If the NotifyClose statement is absent the notification will not be sent. Any of the following will be interpreted as the absence of the statement:

```
xxxxxCode=abcdf  
xxxxxCode=abcdf&NotifyClose=No  
xxxxxCode=abcdf&NotifyClose=n  
xxxxxCode=abcdf&NotifyClose=0
```

If the NotifyClose statement is present the notification will be sent. Any of the following will be interpreted as the presence of the statement:

```
xxxxxCode=abcdf&NotifyClose  
xxxxxCode=abcdf&NotifyClose=Yes  
xxxxxCode=abcdf&NotifyClose=y  
xxxxxCode=abcdf&NotifyClose=1
```

The notification from the client will be in the form:

`info=DocClosed&Stamp=&ServiceID=&DocumentID=&Machine=&Uuid=[&Session=]`

Note : The value '&Session=' will be passed only if cookie authentication is being used.

The PermissionServer should not respond to this notification.

command is invoked (or the document's window is closed), or when the browser in which the document was opened is closed.

DocSaved Notification

The PermissionServer may request to be notified when a local copy of a particular document has been saved-to-disk, by sending the string **NotifySave=<value>** to the client. This request must be made in the RetVal response to the DocPerm request.

The resulting notification is sent to the PermissionServer in the form of a request for offline permission for that document⁴⁵, to which the PermissionServer may respond by giving Offline Perms for the document or not.

DocSaved Notification Values:

The DocSaved Notification is optional, so must be requested. If the NotifySave statement is absent the notification will not be sent. The presence/absence of the request will be determined using the same logic as shown under NotifyClose, above, except the search will be for the term **NotifySave**.

The notification from the client will be in the form:

Request=SaveOffLinePerms&Stamp=&ServiceID=&DocumentID=&Machine=&Uuid=[&Session=]

Note : The value '**&Session=**' will be passed only if cookie authentication is being used.

And the server's response, if positive, should be in the form

RetVal=2[&OfflinePrints=][&OfflineExpire=]&ServId=&Doculd=&Perms=

Where the values for OfflinePrints and OfflineExpire are optional.

And the server's response, if negative, should be in the form:

RetVal=0&Error=<ErrorString>

⁴⁵ In principle, this request should be in the form of a "SavePerm" request - i.e. for permission to save the local copy, as the PrintPerm request asks for permission to print. However, different versions of the Acrobat viewer operate differently in this regard, and it is not currently possible to have the server control saving of a local copy. So the PermissionServer can either prevent saving of a local copy (by sending perms without Save), or can control whether that copy is delivered with offline permissions via this notification.

Document Print Control

When giving Online permission to open a document, the PermissionServer must either grant or deny permission to print that document. If permission is denied, the user will not have access to the File>Print menu or Print icon of the Adobe viewer. Permission to print may be granted in either of two modes:

- **PrintMonitoring:**

Behavior: Under PrintMonitoring, invoking the File>Print menu or pressing the Print toolbar button displays the Acrobat print dialog and printing is performed by the Acrobat viewer⁴⁶.

When the print dialog is dismissed the client monitors the print process and sends the requested notification, if any, to the PermissionServer.

Method: The PermissionServer may request PrintMonitoring by sending the message **AcroPrntDlg=True** in the RetVal response to the DocPerm request. The PermissionServer may also, optionally, request PrintNotification by sending a **Pnot=** instruction in response to the DocPerm request to define the type of PrintNotification to be returned.

- **PrintControl:**

Behavior: Under PrintControl, invoking the File>Print menu or pressing the Print toolbar button displays the FileOpen print dialog and printing is controlled by the client.

When the print dialog is dismissed the client sends a PrintPerm request to the PermissionServer and allows printing based on the response to that request. If printing is allowed, the print job is initiated and monitored; upon completion the client sends the requested notification, if any, to the PermissionServer.

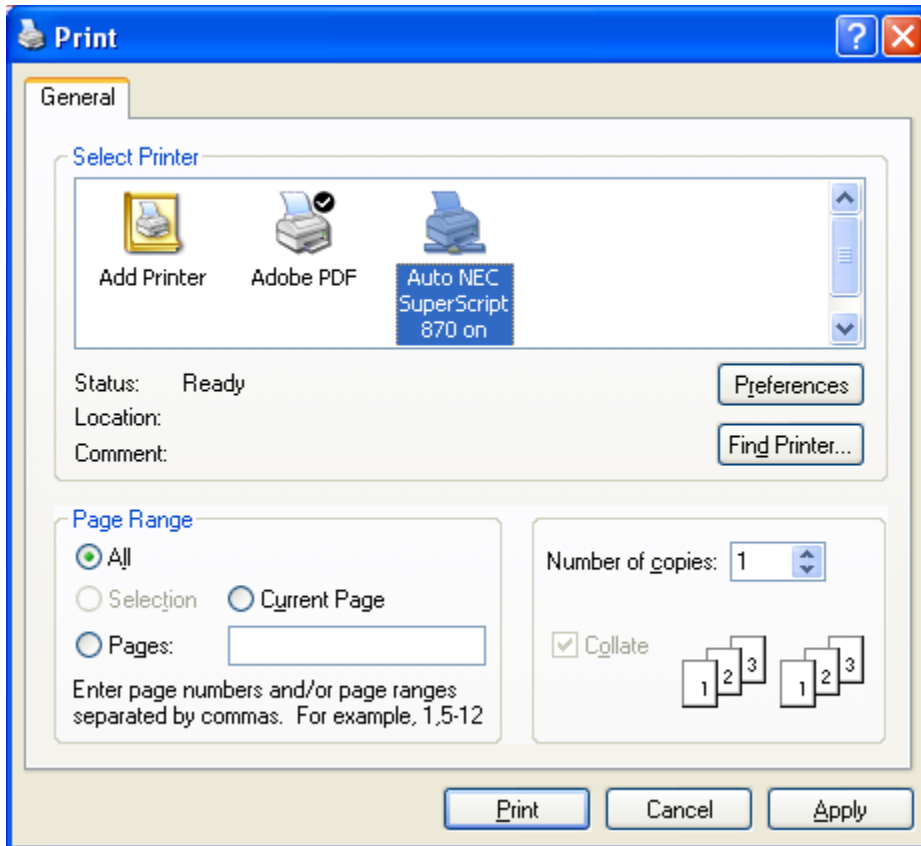
Method: PrintControl is the default state for documents opened via Online access, when print permission has been granted. In addition to controlling the print process via the PrintPerm request, the PermissionServer may optionally request notification of the by sending a **Pnot=** instruction either in response to the DocPerm request or the PrintPerm request.

Note that printing may also be controlled, though not monitored, in Offline mode, via the mechanism described below.

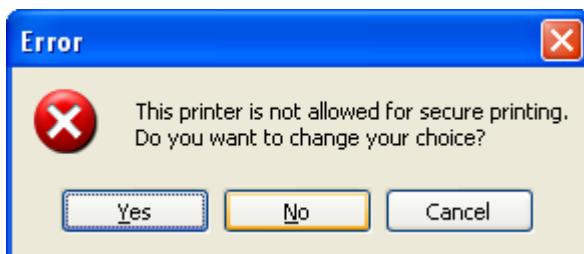
⁴⁶ Under PrintMonitoring neither the FileOpen client nor the PermissionServer can control the user's choice of printer. It is therefore possible that the user will print to a driver that will re-create the PDF file in unencrypted form. Note that different versions of the Adobe viewer provide varying degrees of control over printing to PDF: version 4 products will generally permit such printing, while version 5 and 6 products generally will not.

The PrintPerm Request

If the PermissionServer has responded to the DocPerm request with **Perms=** value that includes permission to print, and has not requested PrintMonitoring, the client will present the FileOpen print dialog in response to the user invoking the File>Print menu or clicking the Print button. The user may then select a printer, some print parameters, and one or more ranges of pages to print, via the dialog shown below.



Note that the dialog appears to allow the user to select the Adobe PDF printer. However, the attempt to use a proscribed printer results in the error message shown below.



Once the user has filled-in the Print dialog and clicked "Print" the print parameters will be routed to your server via the PrintPerm request. This request will be in the form:

`Request=PrintPerm&Stamp=&ServiceID=&DocumentID=&Machine=&Uuid=&ProdVer=&EncrVer=&Count=&PageRanges=&Printer=&AcroVersion=&AcroProduct=&AcroCanEdit=&InBrowser=&DocIsLocal=`

Where:

`Count=` is the number of copies selected by the user

`PageRanges=` is the set of pages the user has requested to print⁴⁷

`Printer=` is the name of the printer driver the user has selected⁴⁸

And the Server's response, if positive, is of the form:

`RetVal=1&ServId=&DocId=&Perms=`

Where:

`RetVal=1` is the return value, 1 for permission

`Perms=` – is the number of copies for which permission to print has been granted (if 0, no printing will be allowed; if 1, one printout, etc.)

And if negative, is of the form:

`RetVal=0&Error=<SomeString>`

Where:

`RetVal=0` is the return value, 0 for error

`Error=<SomeString>` is the reason for denial (this may be any string. The text received will be shown to the user in a messagebox.)

If the PermissionServer has requested PrintNotification, the client will send that notification after the RetVal response has been received, the print job has been sent to the printer and the Acrobat application has notified the plug-in that its print process is complete.

⁴⁷ Page ranges are given in the form `<number of ranges><range>*`. Hence, if the user requests to print page 3 and pages 6 to 8 the list is: 2,3,3,6,8. That is, the request is to print 2 ranges, the first is (3 to 3) and the second is (6 to 8).

⁴⁸ Note that this string will contain escaped characters, e.g. "NEC%20SuperScript%20870"

Document Text Copy Control

When giving Online permission to open a document, the PermissionServer must either grant or deny permission to copy text from that document⁴⁹. If permission is denied, the user will not have access to the menu, toolbar and right-click methods of copying text. Permission to copy text may be granted in any of three modes:

- **Unrestricted**

Behavior: In this mode the user may copy text without server notification or control.

Method: The PermissionServer may grant unlimited copying by returning a Perms value containing the pdPermCopy bit.

- **CopyNotification:**

Behavior: Under CopyNotification the user may highlight text and invoke the Edit>Copy command or equivalent⁵⁰ to copy that text to the clipboard. If the server has requested CopyNotification, the PermissionServer will be notified of the user's copy actions via the Info=WordsCopied notification.

Method: The PermissionServer may request CopyNotification by sending the message **CopyNotify=Yes** in the RetVal response to the DocPerm request.

If CopyNotification is requested, the client will send a notification to the PermissionServer each time the user copies text from the document. This notification will be in the form:

Info=WordsCopied&Stamp=&ServiceID=&DocumentID=&Machine=&Uuid=
&CopySelected=&CopyPageFrom=&CopiedTotal=

Where:

CopySelected= is the number of words selected for copying, and

CopyPageFrom= is the page (in base 1)⁵¹ of the first character of the selection

CopiedTotal= is the total number of words copied (not including the current selection)

⁴⁹ Via the pdPermCopy bit, as described under DocPerm Values, above.

⁵⁰ Among these, on Windows, are Ctrl+C and right-mouse-click>Copy to Clipboard.

⁵¹ That is, counting the first page of the document as page 1.

Note that with CopyNotification the server may not stop copying once permission has been granted. This is possible only with CopyControl.

- **CopyControl:**

Behavior: Under CopyControl the user may highlight text and invoke the Edit>Copy command or equivalent to copy that text to the clipboard, but the text will only be copied if either the number of words is less than that specified by the server as the maximum permitted without permission or the server grants permission for the copy action.

Method: The PermissionServer may request CopyControl by sending the message **CopyMax=<value>** where value is an integer greater than zero. This value will then be used as the threshold for making a server request⁵². In addition, the PermissionServer may add the message **CopyMaxAction=<action>** where <action> is one of those specified⁵³ and the PermissionServer may also add the **CopyNotify=Yes** statement in order to request both CopyControl and CopyNotification, which may be used simultaneously.

When CopyControl has been requested and the user attempts to copy a number of words exceeding that value of CopyMax, the client will make a request to the server as follows:

**Request=MaxWordsCopied&Stamp=&ServiceID=&DocumentID=&Machine=&Uuid=
&CopySelected=&CopyRequired=&CopyPageFrom=&CopiedTotal=**

Where:

CopySelected= is the number of words selected for copying

CopyRequired= is the minimum number of words to satisfy the client copy request⁵⁴.

CopyPageFrom= is the page (in base 1) of the first character of the selection

CopiedTotal= is the total number of words copied (not including the current selection)

⁵² If the number of words selected for copying by the user is less than the value of CopyMax, the copy action will be allowed without any call to the server. If the number of words selected is equal to or greater than CopyMax, the client will request permission for the copy action.

⁵³ If "None" or "0" no action is performed and no additional copying permitted. If "Ask" or "1" the client will make a MaxWordsCopied request when the MaxWordsCopy limit is reached. If "Close" or "2" the client will close the document when the MaxWordsCopy limit is reached.

⁵⁴ This is used only in the MaxWordsCopied request, not in the WordsCopied notification.

The PermissionServer may respond to this request in any of the following ways:

- It may deny permission to copy the text, by returning CopyMax=0. This response may also include an explanation, given via the Message= instruction. For example:

RetVal=1&...&CopyMax=0&Message=Maximum reached!

Or:

RetVal=1&...&CopyMax=0&CopyMaxAction=Close&Message=Maximum reached. The document will be closed!

- It may grant the copy request without changing the maximum number of words that may be copied in subsequent attempts, by returning RetVal=1 and the existing value for CopyMax. Note, however, that in order to permit copying, the response must include some value for CopyMax except in the case where MaxActionCopy is equal to "Close" (in which case the document will be closed immediately). For example (assuming that MaxWordsCopy was set to 50):

RetVal=1&...&CopyMax=50⁵⁵

Or:

RetVal=&...&CopyMax=50&CopyMaxAction=Ask&Message=Your may copy 50 more words from this document.

- It may grant the copy request and modify the number of words the user is allowed to copy, but retuning RetVal=1 with a modified value for CopyMax or a value for CopyAdd⁵⁶. For example:

RetVal=1&...&CopyMax=450 CopyMaxAction=None

Or:

RetVal=1&...=&CopyAdd=400

⁵⁵ This would allow the user to copy 50 more words, then disable further copying - i.e. is the same as CopyMaxAction=None.

⁵⁶ Note that if MaxWordsCopy and AddWordsCopy are defined in the same response, the MaxWordsCopy value used is 'MaxWordsCopy + AddWordsCopy'

Approaches to Copy Control

A PermissionServer may manage copying of text in several ways, among these are:

A) Without checking the number of copied words. Set *CopyMax* to 0 and *NotifyCopy* to true: for each copy attempt the client will send a notification (which is faster than a request) and the PermissionServer will know how many words are selected and the page of the first selected word. This solution is the fastest but because there is no feedback from the server to the client the copy may not be stopped.

B) Checking the number of copied words. Set *CopyMax* to 1, *CopyMaxAction* to Ask and *NotifyCopy* to false: for each copy attempt the client will send a request, and the PermissionServer will know how many words are selected and the page of the first selected word. The PermissionServer may then build and send a response with *CopyAdd* set to the number of selected words (so that the client is allowed to copy the selected words). When the client receives the response it will compute the new maximum, then allow the copy. The PermissionServer may stop the copying of words may in response to any subsequent *MaxWordsCopied* request, by setting *CopyMax* to None or Close.

C) Checking the number of copied words(2).. Set *CopyMax* to a number (say 1000), *CopyMaxAction* to Ask and *NotifyCopy* to true: for each copy attempt the client sends a notification like in approach A. When the *CopyMax* value is reached the client sends a request like in approach B. The coping of words may be stopped when by the PermissionServer setting *CopyMaxAction* to None or Close, but with a resolution of *CopyMax*. This solution is faster than B but has coarser granularity.

Granting Offline Permission

Publishers may wish to allow some or all users to open documents without requesting the decryption key from the publisher's server. Such users are said to have "Offline permission".

The system supports two modes of offline permission, and two delivery methods. The modes are:

- **Classic:** in which the Offline Permissions are given precedence. If Offline Permission is present and not expired/consumed, that permission is used. In this mode users must have an internet connection when downloading the initial permissions, but need never have an internet connection to open or use the files.
- **Inverted:** aka "Auxiliary", in which the Offline Permission is used only if needed. In this mode the client checks whether the local machine has an active internet connection and uses the Offline Permission only if no connection is found.

The delivery methods are:

- **Dynamic:** in which the Offline Permission is given for one file at a time, when the user opens that file with Online Permission. The PermissionServer can deliver these perms by sending RetVal=2. The above inverted mode may be requested by adding OfflineMode=inverted.
- **Static:** in which the permission is given for a group of files. This also may be done in two ways:
 - **PermissionPDF.** When the client encounters a PDF encrypted with DocID=0 it does not make the normal DocPerm request, but rather makes a FilePerm request. The server may then respond with a list, in the form given below, of files to which the user should get offline permission.
 - **Request:** Instead of creating the PermissionPDF, the server may at any time (e.g. in response to a request=settings or request=DocPerm) send the instruction RequestFilePerm=true. This will instruct the client to make a FilePerm request, as if it had opened a PermissionPDF.

Details of Offline Permission Modes

Offline permission is granted to a specific machine and is exclusive to that machine. As noted, Offline permission may be given in two modes, ***Static Offline Permission*** and ***Dynamic Offline Permission***.

Static Offline Permission is granted to a list of documents, which must be specified by the PermissionServer in the form of an offline permission list then sent to the client in response to the *FilePerm* request or a *RequestFilePerm* instruction. The publisher may assign additional restrictions, e.g. an expiration period, to the offline permission in general or to specific documents within the offline permission structure.

Dynamic Offline Permission is given for a particular document, in response to the user's request to open that document via the *DocPerm* request.

When the user attempts to open a document from local storage the FileOpen client performs the following steps:

1. Retrieves from the **ServiceID** and **DocumentID** from the document.
2. Checks for the presence of a file, **permissionFile**, with a name containing the **ServiceID**, in the designated storage location⁵⁷. If this file is not present, the user does not have offline access and permission to open the document must be obtained via the client-server interaction.
3. If the permissionFile is present, it is searched for an entry corresponding to the document's DocumentID and then the pre-defined fields described below.
4. Two general states are supported:
 - Mode is Inverted:
 - i. If mode is inverted, the client first checks to see whether the local machine has an active internet connection. If so, the client uses the client-server permission mechanism. If not, the client uses the Classic Offline Permission mechanism.
 - Mode is Classic:
 - i. If Inverted mode is not specified, the client supports two states:
 - Permission is present (there is an entry for the DocumentID and a key with the decryption key as its value and other entries, if any, and the permissions have not expired or been consumed.)
 - ii. Result: the document is opened
 - Permission is Undetermined (the file does not exist or there is no entry for the DocumentID)
 - iii. Result: client will perform the online authorization.

⁵⁷ For example, the folder C:\Documents and Settings<username>\Application Data\ FileOpen\

Creating and Updating the Offline Permission List

The PermissionServer is responsible for assigning and monitoring the offline permission status of users. When a user has been granted Offline Permission rights or the number of files for which that user has Offline Permission has changed – i.e. differs from the number of files for which that user has already downloaded permission - the publisher's server should invoke the RequestOfflinePerms mechanism or send the user to some kind of **Offline Permission page**. This page should be, or include a link to, a document initiating the update of offline permissions, as described in the next section.

Mechanism for Granting Static Offline Permission

If the user elects to download the new or updated permissions the server should present the user with a standard PDF, the **permissionPDF**, encrypted with a semantically meaningful DocumentID. The permissionPDF must have DocumentID of zero, expressed numerically ("0"), and must be encrypted using the key "perms" (without the quotation marks). The PermissionPDF must also be encrypted using the same ServiceID as the documents for which offline permission is being provided.

The client will then request permission to open a document with this DocumentID, as follows:

If a user to whom offline permission has not been granted attempts to open the permissionPDF the server should respond with a message string stating the reason why permission was denied; this string will then be presented to the user in the normal manner.

If the user requesting offline permission is entitled to such permission, the publisher's server should respond with an acknowledgement of permission followed by an offline permission file. This response should be of the form:

RetVal=1&File=

or

RetVal=0&Error=

Where:

RetVal= is the positive or negative response to the offline permission request.

File= is followed by the contents of the offline permission file, given below, or

Error= is followed by a string explaining the reason for denial (this string will be presented to the user)

Note: the uniqueID of the permissionPDF should never be included in the list of documents in the offline permission file (as this would allow the client to open that document locally, negating the update process).

Structure of the Offline Permission File

The **Offline Permission File** is organized using a protocol⁵⁸ intended to minimize the size of the file and to remove the possibility that a user will be prevented from opening a document to which the publisher has granted access when that user has a valid and up-to-date Offline Permission File.

The Offline Permission File has three logical sections:

- The **Header**, containing data about the file, including the ServiceID the UserName and default values.
- The Content, containing a list of the UniqueID of each document for which the user either has offline permission or has no permission (neither offline nor online permission).
- The **Trailer**, containing an entry specifying the number of items in the Content section.

The structure of the Offline Permission File should be as follows (description following “//” must not be included in response, mandatory values are in red):

[Header]	
Service =594	//the ServiceID ⁵⁹
Perms=1	//usage permission ⁶⁰
PrtCnts=1	//the number of printouts allowed ⁶¹
Expire= 2006/11/01 00:00:00	//the date of expiration (see below)
Date= 2006/01/01 00:00:00	//date the file was generated (GMT) ⁶²
Action=New	//the action statement, if any (see below)
Key =abcde	//the global decryption key, if any
Password=	//the password, if any (default is no password) ⁶³
OpnCnts=	//the number of times the document may be opened
OfflineMode=	//the type of Offline Permission ⁶⁴
BlockScreenShot=true	//The default is to not block screen capture.
Password=	//The offline password, if any
[UniqueID1]	//UniqueID of a document
Key=1a2K3	//decryption key for that document

⁵⁸ Documents should be Included In the list only when the user's permission is explicitly stated. That is, the Offline Permission File should include those documents for which the publisher has granted offline permission. The list should not include those documents for which online permission is required.

⁵⁹ This may be any string of up to 63 characters.

⁶⁰ Default is 0 (no permission). Documents won't open if this key is not present and set to a value that permits opening.

⁶¹ If this entry is present, printouts will be limited to that number. The absence of this entry is interpreted as "unlimited printing". A value of 0 is interpreted as "no printing", thus is equivalent to a Perms= statement in which printing is not allowed – if you wish to forbid printing you should do so via the Perms value.

⁶² This should be either a well formed 'military' date, as above, or "now" without the quotes.

⁶³ If a password is specified the user will be asked for that password when attempting to open files using the Offline Permissions.

⁶⁴ The default, e.g. if this term is missing, is "Classic". To set Inverted Offline Permission specify OfflineMode=Inverted or OfflineMode=I; see "Inverted Offline Permissions", below.

Expire=2002/01/01 00:00:00	//expiration period, if not default
Perms=68	//permission, if different from global value
Expire=2006/02/01 00:00:00	//Expiration, if different from global value
PrntCnts=4	//printouts, if different from global value
OpnCnts=2	//open count, if different from global value
OfflineMode=	// type of Offline Perms, if different from global value
BlockScreenShot=false	//screen block, if different from global value
[UniqueID2]	//UniqueID of a document
Key=2x5yG	//decryption key (required if no global key)
[UniqueID3]	//UniqueID of a document
[UniqueID4]	//UniqueID of a document
[Trailer]	//last entry
#docs=4	//total number documents in list

The presence of a document's UniqueID in the list without the presence of the corresponding decryption key means that the system must use the Global Key, defined in the Header. So in the above example the documents with UniqueID3 and UniqueID4 will inherit all the Global permissions given in the Header, while those with UniqueID1 and UniqueID2 will have the permissions assigned to those specific documents.

Also note that the document with UniqueID1 will expire on a date different from the default and has permissions different from the default, while the document with UniqueID2 may be opened with the default permissions until the default expiration date.

Offline Expiration

The default offline expiration state is none. That is, if the Expire= statement is absent or has the value "never" without the quotation marks, expiration will not be imposed.

When used, the offline permission structure supports two types of expiration: **Absolute** and **Relative**.

As with other parameters, the scope of the expiration statement may have either **Global Scope** (the value given will be applied to all documents in the list) or **Local Scope** (the value given applies to only the document for which it has been specified, and overrides any Global permission for that document). The scope of parameters is given by their location in the Offline Permission File: Expiration statements in the Header section have Global Scope; those under a DocumentID have Local Scope for that document.

Both Absolute and Relative expiration must be prefaced by the Expire= statement. The two formats are as follows:

Absolute Expiration is given as a time value in Greenwich Mean Time (GMT), this value should be expressed in the format yyyy/mm/dd hh:mm:ss – with the hours given in military time, e.g. 22:01:00 represents 10:01 PM and 02:01:00 represents 2:01 AM.

The following is a valid expression for Absolute Expiration on November 12, 2006 at 5:30 PM.

```
Expire=2006/11/12 17:30:00
```

Relative Expiration is given as a pair in the form <number><name>, where <name> is a string of which only the first character is significant. There must not be a space between the number value and the name value.

The expected values for <name> are:

- **d** (i.e. days)
- **m** (i.e. months)
- **y** (i.e. years)

Hence the following are valid expressions for Relative Expiration:

```
Expire=3d  
Expire=2m  
Expire=1y
```

The following formulations are also valid, though not recommended:

```
Expire=3days or Expire=3day or Expire=3ds  
Expire=2months or Expire=2monthlyperiods or Expire=2mos  
Expire=1year or Expire=1yr
```

A statement of the form Expire=3 d (i.e. with a space between 3 and d) is not valid.

Note that the Relative Expiration Date is relative to the time given in the header as 'Date' (not to the date of the offline permission file creation or update).

The Action= Statement

The statement 'Action' may be inserted into the header part of Offline Permission File to modify the behavior of the client as it processes offline permission files. This statement is interpreted to determine what the client should do with the file being received. The following values are supported:

Action=New: a new permission file is created for the service (the previous one is deleted)

Action=Update: the current file is updated to reflect the new permissions (or is created if missing).

If no value is provided for the 'Action' statement, i.e. that statement is missing, 'Action=New' is assumed for backward compatibility.

A statement of the form Action=Update may be followed on some fields by a qualifying statement 'All'. If used, the All statement should follow the Update statement after a comma without any space, as follows:

Action=Update,All

The All statement has global scope and precedence. That is, when a value 'All' is encountered in the header of the Offline Permission File, the values in the Header section of the Offline Permission File will be applied to every document in the user's Offline Permission File. If the user has documents in the Offline Permission File with specific permissions (i.e. permissions applying only to that document, overriding the global permissions in the original Header), those permissions will be incremented by the new permissions provided by the Update statement.

For example, if the Offline Permission File contains in the Header the statements:

```
Action=Update,All
PrtCnts=nnn
```

Or contains in the Header the statements:

```
Action=Update
PrtCnts=nnn,All
```

Then for every document in the existing Offline Permission File with PrtCnts=mmm, the number of prints will be increased by nnn so that PrtCnts=(nnn+mmm).

The general rule governing computation of an updated print counter is:

- if the current counter is unlimited (-1), the updated value is the new counter as provided in the Offline Permission File containing the Action=Update statement.
- if the current value is *not* unlimited, then:
 - if the new counter is unlimited, the updated value is unlimited
 - otherwise, the updated value is the current counter + the new counter

Note that the meaning of the All statement depends on which field it is modifying, as follows:

- *Action: 'All'* means that the 3 following fields will be updated

- *Expire*: 'All' means that all expiration dates will be updated
- *Perms*: 'All' means that all perms will be updated
- *PrtCnts*: 'All' means that all the print counters will be updated

Hence a statement of the form:

`Expire=3d,All`

Would result in all documents in the Offline Permission List being updated with a three day expiration.

And a statement of the form:

`Perms=5,All`

Would result in all documents in the Offline Permission List being opened with permission to Print; note that in this absence of a `PrtCnts=` statement, printing would be unlimited.

Mechanism for Granting Dynamic Offline Permission

Offline permissions may also be granted for a particular document when giving online permission. In this case, rather than returning the standard response to the DocPerm request, in the form:

`RetVal=1&ServId=&Docuid=&Perms=&Code=`

The server should return a response granting Dynamic Offline Perms, as follows:

`RetVal=2&ServId=&Docuid=&Perms=&Code=&OfflinePrints=&OfflineExpire=&OfflineOpens=`

Where:

`RetVal=2`: is the statement signifying that the document should be added/updated into the Offline Permission File

`OfflinePrints=`: is the statement signifying the number of `PrtCnts` to grant for that document.

Note that this value is *optional*. If missing the default print count from the Offline Permission File, if any, is used

`OfflineExpire=`: is the statement signifying expiration period to grant for that document.

`OfflineOpens=`: is the statement signifying the number of times the document may be opened.

The value following `OfflinePrints=/OfflineOpens=` and `OfflineExpire=` should be in the format (c (count) and d (date), respectively) defined for Offline Permission Files ('unlimited', -1 or a positive integer for c, 'never', xmonths, ydays or well formed date for d).

Inverted Offline Permissions

In the default mode, Offline Permission is given precedence over online permissions. That is, if a machine has Offline Permission for a given document and that permission is not expired or consumed, the client will use the Offline Permission whether or not the machine has an active internet connection.

With client 0709 or later the PermissionServer may also provide Offline Permission without precedence, in a mode that instructs the client to use the Offline Permissions only if the machine is not connected to the internet. This mode is known as “Inverted” Offline Permission.

Inverted Offline Permissions may be given as part of the Dynamic Offline Permission instruction, using the following syntax:

```
RetVal=2&ServId=&Docuid=&Perms=&Code=[&OfflinePrints=&OfflineExpire=&OfflineOpens=]  
&OfflineMode=<mode>
```

Where <mode> may be:

'c[lassic]', 'C[urrent]', 'n[ormal]', 'N[yyyy]' or '0' for the default classic mode. The absence of the OfflineMode= statement will produce the same result as this response.

'i[nverted]', l[versed] or '1' for the inverted mode.

Additionally, the OfflineMode= statement may be included in the response to a Static Offline Permission request (Request=FilePerm&...). Here the OfflineMode=Inverted statement must be included into the File= response, where it may be given Global Scope or Local Scope, as defined above.

Note that the OfflineMode= statement is not recognized by earlier versions of the client, so will generate a parser error is served in a FilePerm response to any client with build number less than 709. You must therefore parse the Build= statement in the FilePerm request and only respond with this value if Build=709 or higher.

Local Offline Permission Value Reporting

In order to enable the server to maintain a single count of both Online and Offline Permission values the client sends the value of the Offline Permission for a given document with some Online Permission

requests concerning that document. These values may then be modified to reflect the Online Permissions granted in the current session, then returned to the client for storage⁶⁵.

Three pairs are employed, with the following values:

OfflineExpire Ascii GMT time (Military time)

OfflinePrints Number of copies remaining in the offline permission file

OfflineOpens Number of opens remaining in the offline permission file

The pairs are sent in the DocPerm, FilePerm and KeepOpen requests (they are also sent with the Setting request, but as that request is considered to be generic - i.e. applies to all documents in the session – it is recommended that the values in the DocPerm be used). The values are not sent in PrintPerm request though the OfflinePrints value will be added to this request in a future client version.

The three OfflineXxxx values are sent only if they are known, that is:

- there is an offline permissions file containing an entry for the document in question
AND
- a server request is done, because mode is:
 - o Inverted (the offline permissions are known but not used)OR
- o Normal AND a required offline perm has reached 0 (Online Permission is needed).

Offline Failure Notification

In the event that the client makes an online permission request because Offline Permission was expired or consumed it will include values in the request to notify the PermissionServer of that fact. This data may be useful in identifying the cause of an unexpected request

This data is given by the term OfflineFail= followed by a value taken from a bitset containing values from 0 to 7. The supported values are:

⁶⁵ For example, imagine a server tasked with maintaining a single total of Online/Offline usage (e.g. a total of five document opens, whether these are performed Online or Offline). That server must parse the values uploaded in the DocPerm requests as OfflineOpens. That is, if the User was granted OfflineOpens=5 then went offline and opened the document two times, the next time that User attempts to open the document while connected the DocPerm request will contain OfflineOpens=3 and the PermissionServer, if granting the request, should then return OfflineOpens=2, which will be written to the Offline Permission.

- OfflineFail=0 No failure (in this case the term will not be included in requests)
- OfflineFail=1 The permission file or permission record time is expired
- OfflineFail=2 The open count permission has been consumed (no more opens allowed)
- OfflineFail=4 The print count permission has been consumed (no more prints allowed)

The bitset permits additive values, e.g. OfflineFail=3 means that the time is expired (file or document record) and also that the open count has been consumed.

Note that expiration values are always checked first and if it is "expired" the file or the record is NOT verified. That is, for a print permission whose time is expired and whose print count is finished, the OfflineFail value will be 1 and not 5.

Location of Offline Permission File

When Offline Permission is granted to a machine the data is stored in a file written to a file in the location specified by the following:

Windows:

C:\Documents and Settings\<login>\Application Data\FileOpen\<DataFolder=>\<USR><ServiceID>.sek

Where:

<login> is the user's login

<DataFolder=> is the value given in the .foe file used to encrypt the document

<USR> is the number of the license used to encrypt the file

<ServiceID> is the value of the document ServiceID

Macintosh:

Dynamic Forms Manipulation

The system supports dynamic insertion of text into a document via an instruction from the server. This mechanism may be used to perform a type of real-time watermarking on documents, e.g. insertion of the user's name or other data onto the page, when the document is opened and/or printed.

To implement this method the document must contain a text-only form field prior to encryption⁶⁶. There is presently no automated mechanism for insertion of this form field, but commercial tools with the capability are thought to exist.

When a document is opened and/or printed in Online Mode⁶⁷, the server may add to the DocPerm or PrintPerm response elements that inform the client of the name of the form field to manipulate, and the text to insert into that form.

The server may send a response in the following form:

...&FormField=FirstFormField=Updated text for first field&FormField=SecondFormField=Updated text for the second form field&...

As noted, this data may be sent in response to the DocPerm request, in which case the new text will be displayed on the visible pages of the document, or in response to the PrintPerm request, in which case the text will appear on the printed page.

⁶⁶ There may be up to five form fields on each page, and each form field may contain up to 1023 characters. Control characters like '\n', '\t' should be properly managed by Acrobat.

⁶⁷ Presently this capability is supported only in Online Mode.

Use of the System in an Intranet Environment

The WebPublisher client may be configured to operate in an Intranet environment. This configuration is obtained from the Windows registry settings on the local machine. When the proper settings are detected, the client will not contact the corporate proxy server when accessing the URL specified as the target of the DocPerm, FilePerm and PrintPerm requests.

In order to operate in Intranet mode, the target URLs must be identified in the local machine registry as local servers. The logic used by the client to determine whether a server is local is as follows:

- It first checks the IP number to see if it is one of the reserved local IP addresses (10.x.x.x, 172.16.x.x to 172.32.x.x, 192.168.x.x)
- Then it searches in the exception list. The search follows the Microsoft documentation:
 - The proxy bypass list contains one or more server names separated by semicolons or whitespace.
 - ([<scheme>=][<scheme>:"//"]<server>[":"<port>])
 - The proxy bypass list can also contain the string "<local>" to indicate that all local intranet sites are bypassed.
 - Local intranet sites are considered to be all servers that do not contain a period in their name.
 - You can list locally known host names or Internet Protocol (IP) addresses in the proxy bypass list.
 - This list may contain wildcards, such as "*", that cause the application to bypass the proxy server for addresses that fit the specified pattern, for example, "*.microsoft.com" or "*.org".
 - Wildcard characters must be the left-most characters in the list. For example, "aaa.*" is not supported.
 - To list multiple addresses and host names, separate them with blank spaces or semicolons in the proxy bypass string.
 - If you specify the <local> macro, the function bypasses any host name that does not contain a period.

The Sample Scripts

The WebPublisher distribution contains a sample server-side application which demonstrates the basic functionality of the system. These scripts are presented in two forms, in Perl (.pl) and in ASP (.asp). The two scripts are identical in functionality. Operating versions of these scripts may be found at the following locations:

Perl: <http://www.fileopen.com/fowp>

ASP: http://66.70.52.14/fowp/index_asp.htm

The functionality of the scripts is as follows:

Fowp.* – Demonstrates the receipt and return of messages using the above Communications Protocol.

Fowpsesscook.* – Demonstrates the placement of a session cookie onto the user's machine, in a format which is recognized by the Client.

Fowpusercook.* – Demonstrates the placement of a user cookie onto the user's machine, in a format which is recognized by the Client.

The fowp.* script implements a simple and hard-coded mechanism for granting permission, based on the value of the DocumentID. The behavior of the script can be seen by opening the demonstration html pages. In this page, documents are identified by DocID, and the response of the fowp.pl is keyed to those DocIDs. Example files are also included to demonstrate the provision of Offline Permission.

Troubleshooting

Please send comments and suggestions to sbingham@fileopen.com

Appendix 1: Document and User Identification Constants

DocumentID

Usage: Required

Format: a string of not more than 63 printable characters.

Function: Used to identify the document. Typically, though not necessarily, unique.

Syntax:

In .foe file: "**DocumentID**=SomeString"

In commandline: "**-i** SomeString" or "**--DocID** SomeString"

In DocPerm Request: "**DocumentID**="

Required In RetVal: Yes

In PrintPerm Request: "**DocumentID**="

Required In RetVal: Yes

ServiceID

Usage: Required

Format: a string of not more than 63 printable characters.

Function: Typically used to identify the group of which the document is a member, though may be used to store any data.⁶⁸

Syntax:

In .foe file: "**ServiceID**=SomeString"

In commandline: "**-s** SomeString" or "**--SerID** SomeString"

In DocPerm Request: "**ServiceID** ="

Required In RetVal: Yes

In PrintPerm Request: No

Ident3ID

Usage: Optional

Format: a string of not more than 63 printable characters.

Function: May be used to store any data.

Syntax:

In .foe file: "**Ident3ID**=SomeString"

In commandline: "**-3** SomeString" or "**--Id3ID** SomeString"

In DocPerm Request: "**Ident3ID** ="

Required In RetVal: Yes, if specified at encryption

In PrintPerm Request: No

⁶⁸ The ServiceID is used by the client as part of the name for the Offline Permission file.

Ident4ID

Usage: Optional

Format: a string of not more than 63 printable characters.

Function: May be used to store any data.

Syntax:

In .foe file: "**Ident4ID**=SomeString"

In commandline: "**-4** SomeString" or "**--Id4ID** SomeString"

In DocPerm Request: "**Ident4ID** ="

Required In RetVal: Yes, if specified at encryption

In PrintPerm Request: No

Encryption Key

Usage: Required

Format: an alphanumeric or hexadecimal string of exactly 5 or 16 bytes⁶⁹

Function: Used to encrypt and decrypt the document. Must be stored

Syntax:

In .foe file: "**CryptoKey**=abcde"

In commandline: "**-k** abcde" or "**--Key** abcde"

In RetVal: "**code**="

Authentication Type

Usage: Required

Format: One of several pre-defined constants

Function: Identifies the data to be retrieved by the client and passed to the PermissionServer as part of the permission requests.

Syntax:

In .foe file: "**CryptoKey**=abcde"

In commandline: "**-k** abcde" or "**--Key** abcde"

⁶⁹ Note that the space character is allowed, so a key presented as a string containing a space, e.g. "abcde ", will produce undefined results. The string must be exactly 5 (40-bit) or 16 (128-bit) printable characters or a string of exactly 10 (40-bit) or 32 (128-bit) hexadecimal characters.

Appendix 2: Details of Online and Offline Modes

Because Offline permissions are distributed via an online mechanism, and in the case of Dynamic Offline Permissions are distributed as part of the Online permissioning process, the boundary between Online and Offline permissions is not always apparent. For example, if a PermissionServer distributes a document with online permission to open and print and also gives offline permission to print one time, what should happen when the user attempts to print? Should the client make a PrintPerm request, or use the offline print?

The following statements govern permissions:

- When opening a document, the client searches first for Offline permissions. If Offline permissions are present and have not expired or been consumed and the flag for Inverted Offline Permission has not been set, the client uses these permissions. The client is then in **Classic Offline Mode**.
- If Classic Offline Permissions are not found or have been consumed, the client connects to the PermissionServer; if permission is granted and the document opened, the client is then in **Online Mode**.
- If the flag for Inverted Offline Permission has been set the client first checks to determine whether the local machine has an active Internet connection. If so, the client operates in **Online Mode**. If not, in **Classic Offline Mode**.
- If a document is opened in Offline Mode but the user tries to perform an action for which Offline permissions have been consumed⁷⁰, the client will attempt to get permission for this action from the PermissionServer. In this case, the client is still in Offline Mode, but the PermissionServer may grant or deny permission for the action. The Offline Permission file is not updated.
- If the user opens a document in Online Mode but the server gives Dynamic Offline Permissions, the client goes into Offline Mode. Accordingly, in the example above the user's attempt to print would be allowed on the bases of the OfflinePrints= statement in the RetVal to the DocPerm request⁷¹.

⁷⁰ Current this would apply to opening and printing.

⁷¹ This assumes that the Perms= statement included a values that permits printing.

Appendix 3: Setting up the ActiveX and Java⁷² Installers

The WebPublisher3 distribution contains a mechanism for installing the client from within a user's web browser, provided that the browser supports ActiveX controls. This installer, FileOpen.cab, should be placed onto the web in conjunction with an HTML page containing the code given below.

The goal of the ActiveX install procedure is to permit the PermissionServer to determine which users have which version of the plug-in and hence when the install procedure is required, and, when installation is required, to allow a user to get the plug-in with a minimum of interaction. The logic of determining which version of the client is required, and which is present, is not part of the installer itself but rather must be implemented in the PermissionServer logic that dispenses the installer.

Logic of the Install Procedure

The flow of the install process is as follows:

1. The PermissionServer must first determine the user's Operating System and browser type. The ActiveX install will work only on Windows and only within browsers that support ActiveX controls and have those controls enabled⁷³. Accordingly, the PermissionServer must support at least three possible states:
 - a. The user is on a Mac; the user should be redirected to a page linking to the Macintosh installer.
 - b. The user is on Windows, but is using a browser that does not support ActiveX; the user should be redirected to a page linking to the FileOpenInstaller.exe installer (the **Manual-Install** page).
 - c. The user is on Windows and has a browser that supports ActiveX; the user should be redirected to the page containing the ActiveX installer (the **Auto-Install** page).
2. The PermissionServer should check the local machine for a cookie from its own domain with the name **FileOpenPlugin**. If present, this cookie will contain a string **PIVersion=** followed by a three digit number. This cookie is written by the installer, so the presence of that cookie is a

⁷² Information on the Java installer is available by request.

⁷³ As a general statement, this means Microsoft Internet Explorer only. Other browsers may support ActiveX, but we have tested only IE. We have not yet established a mechanism for determining whether the user's security settings will allow ActiveX controls to run.

- good indication that the plug-in is present⁷⁴. The number corresponds to the Private Build Description value of the version of the plug-in on that machine.
3. If the build number of the installed plug-in is high enough for your application then no install is needed, and the user should be redirected away from the install page.
 4. If the build number is too low, or the cookie is not present, the user should be directed to the Auto-Install page.
 5. Depending on the local security settings, the user will typically see either the Security Warning dialog shown in Figure 1, or will see the ActiveX control shown in figure 2.⁷⁵ Accepting the dialog in Figure 1 will bring the user to the dialog in Figure 2.
 6. When the user clicks on the OK button in the dialog shown in Figure 2, the install process will begin. At this point the code executing is the same as that in the FileOpenInstaller.exe.
 7. Upon completion of the installation procedure, the installer will write a cookie on the machine from the domain specified in the **CookieUrl** parameter of the html file, then will show the user a "Finish" button as shown in Figure 3.
 8. When the user clicks the Finish button the installer will redirect the user to the page specified in the **TargetUrl** parameter of the html file.

⁷⁴ The presence of the cookie is not proof that the plug-in is present, as the user may have removed the plug-in after installation. If this degree of uncertainty is unacceptable, the PermissionServer should skip the check for the cookie and run the ActiveX for all users.

⁷⁵ On machines with security set very high, the user may see no control at all. The explanatory text on this page should tell users that if they see no control they should go to the Manual-Install page. In some other cases, users will encounter an error stating that the FileOpen.api could not be copied; the title bar of this dialog contains an explanatory error number given by the WindowsAPI and defined at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/debug/base/system_error_codes.asp



Figure 1: The Authenticode Signature dialog. This dialog appears when the user's security settings are at the default (Medium). If the user's settings are lower, this dialog is skipped. If higher, the ActiveX control may not be allowed to load, and the user may need to run the .exe installer.



Figure 2: The welcome screen. The number at lower left identifies the version of the installer, and is in the form <installer major version>.<installer minor version>.<plug-in version>

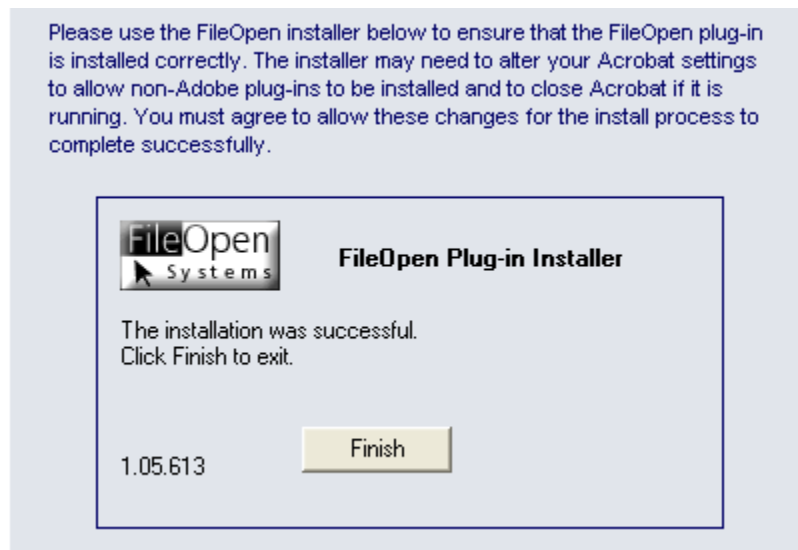


Figure 3: The completion screen.

Entries in the Auto-Install Page

The Auto-Install page must contain entries in the html specifying the version of the installer, the URL to which the installer should send the user upon successful installation, and the domain from which the completion cookie should be written. This syntax is shown below. Portions in **red** should be modified.

codebase="FileOpen.CAB#Version=1,0,0,9"⁷⁶

The TargetUrl parameter:

<PARAM NAME="TargetUrl" VALUE="http://www.fileopen.com/fowp/docs/M-OPQ-D-100.pdf">

The URL given in this parameter will be used by the installer to redirect the user when the install is complete. The target can be any URL, i.e. does not have to be a .pdf file, but is often used to validate the install, e.g. by linking to an encrypted **targetPDF**. If the user can open the targetPDF, the install was successful, and the PermissionServer will be notified that the install was completed via the DocPerm request for the targetPDF.

The value of the parameter may be inserted dynamically, if the page is being created on the fly. Inserting the value dynamically allows the installer to be used in as part of an on-demand system, for example as an intermediate step in an eCommerce transaction in which a user purchases a PDF, that PDF is encrypted dynamically and the URL to the encrypted file inserted into the TargetUrl parameter.

⁷⁶ This number can be obtained from the *File Version* value in the resources of the FileOpen.dll program, which is inside the FileOpen.cab (right click, get properties). If unsure, please contact webpub@fileopen.com to get the current value of this number.

The CookieUrl parameter:

<PARAM NAME="CookieUrl" VALUE="http://www.fileopen.com/">

The URL given in this parameter is used by the installer to create a cookie to mark the machine as one with the plug-in installed. The value of this parameter must be the domain from which the PermissionServer is running, as servers are not allowed to retrieve cookies from other domains.

The structure of the cookie is shown below, with comments following the "//" character.

FileOpenPlugIn	//name of the cookie
PIVersion=704	//value; this is the Private Build Description of the installed plug-in
www.fileopen.com/	//domain for which the cookie was written
3584	//below is timestamp data; the cookie written by the release installer is
3592585216	//set to expire on Fri., 01-Jan-2010 00:00:00 GMT
29683604	
2420869968	
29614142	
*	

Client version

Name of the Product: WebPublisher3 - PDF Encryptor

Copyright: Copyright(c) 2001-2005 FileOpen Systems Inc.

Product version & build. The version numbering is 3 for Fowp3, x.y.z for the build number. Build is a 3 digit increasing number. The last (current) version & builds are: Version '3.7.0.7', Build '707'

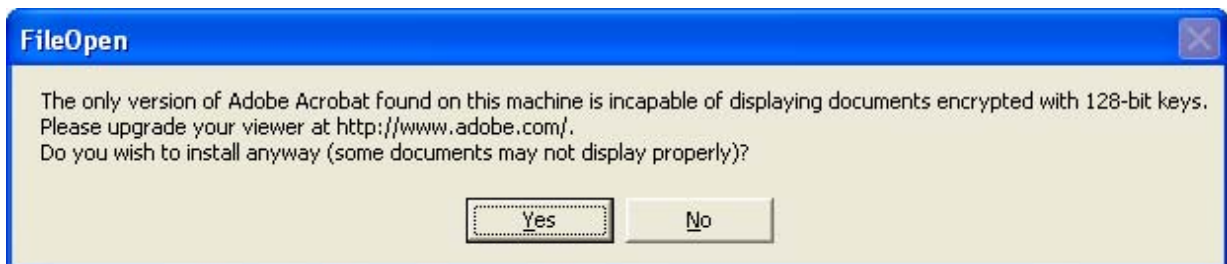
File: Version and date. The file version is: unit of the year, month, day and hour. The last (current) version & date are: Version '5.04.07.22' Date 'April 7, 2005 at 22:00 GMT'

The 128-bit Installer, binary:

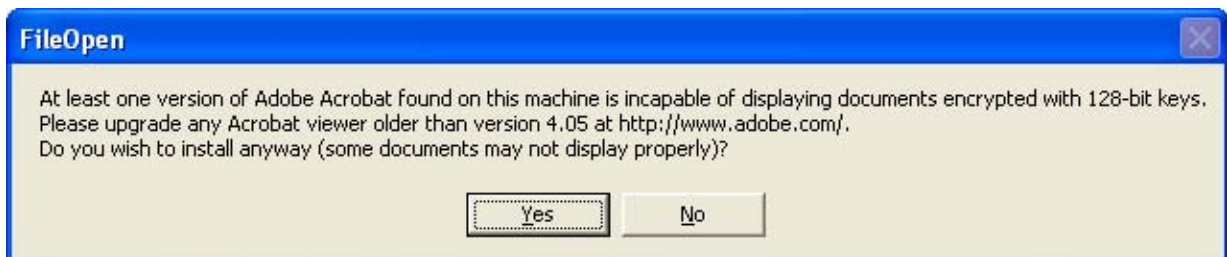
If you have encrypted your documents using 128-bit keys and are distributing documents to users who may have only Adobe Acrobat or Reader 4.0, you should use the installer in 128-bit mode. Acrobat 4.0 and Reader 4.0 support only 40-bit keys, so users must have version 5.0 or later to open files encrypted with 128-bit keys⁷⁷.

The installer will operate in 128-bit mode when renamed to FileOpenInstaller128.exe⁷⁸. In this mode the detects the version number of the local copy of Acrobat and/or Acrobat Reader and presents a message to the user. Two scenarios are supported:

1. The machine contains only one version of Acrobat and that version is insufficient to display documents encrypted with 128-bit keys. In this case the installer will display the following message:



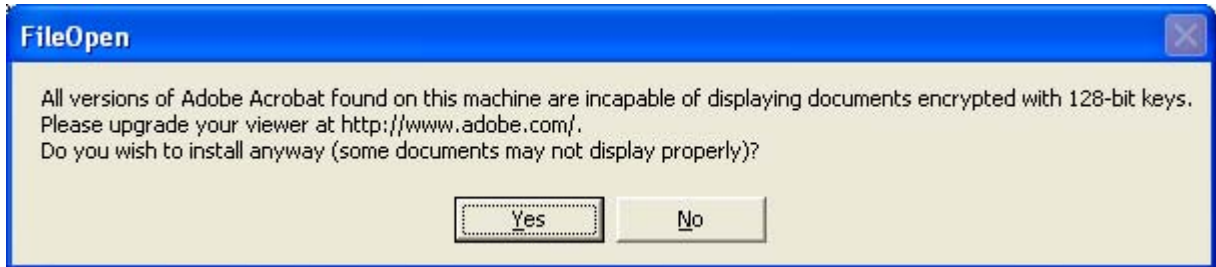
2. The machine contains more than one version of Acrobat and one of these is insufficient to display documents encrypted with 128-bit keys. In this case the installer will display the following message:



⁷⁷ In fact, version 4.0 will *open* files encrypted with 128-bit keys, but the PDF pages of these files will be blank. More accurately, we should say that version 5.0 or later is required to *display* PDF files encrypted with 128-bit keys.

⁷⁸ Or any other name containing the digits

3. The machine contains more than one version of Acrobat and all versions found are insufficient to display documents encrypted with 128-bit keys. In this case the installer will display the following message:



Appendix 4: User Authentication Modes

The fundamental client/server interaction supported by WebPublisher is a request by the FileOpen.api client to the Publisher's PermissionServer and a response with data to grant or deny a user permission to perform an action on a document. By design, the identity of the document is fixed, having been defined by the publisher at the time of encryption. The structure of the requests and responses is defined by the FileOpen protocol. But the identity of the user must typically be derived from data obtained by the client from the local machine context and passed to the PermissionServer in the request.

Accordingly, in cases where the identity of users is significant, the effectiveness of the system depends on the proper selection of one (or more) of the available Authentication modes. This section attempts to provide some guidance in choosing among these modes.

Static or Dynamic Encryption:

The first decision governing user identification is whether to encrypt the files dynamically ("on the fly", typically on a server) or not. If a file is encrypted dynamically for a particular user identity (obtained, for example, from that user's login to the webserver) then that identity may be inserted into the file itself, e.g. in the ID3 or ID4 container. In this case the PermissionServer need not obtain the user's identity via the Authent=m mechanism, as that data is provided by the document data itself.

Encrypting files dynamically is more complex, and requires more resources, than encrypting the same files offline and serving the pre-encrypted files. Publishers must therefore consider the tradeoff between the benefits of this unified user identification mechanism and the added complexity of implementing server-based encryption. If dynamic encryption is used, the Authent=m mechanism is typically employed, as this requires no user intervention.

Online or Offline Access:

Assuming Static Encryption, the choice between Authent= methods should also be governed by the method of file distribution. Not all of the Authent= mechanisms are designed for offline use. If, for example, files are to be distributed on a standalone CD-ROM without integration with a website, the Authent=c (cookie) mechanism is not useful. By contrast, if the files are to be distributed via a website to users who are permitted to login from any machine, the Authent=m mechanism will be insufficient.

Note also that it is no longer a requirement that only a single Authentication method be used. The XAuthent mechanism permits dynamic switching between Authentication modes.

Relative Usefulness of Authentication Methods:

Cookie Authentication:

The single cookie method (Authent=c/CommUse=s) is intended for use with a webserver. That server must deposit a permanent cookie onto the local machine, and the document must contain a description of that cookie sufficient to enable the client to retrieve that cookie.

Pro:

- Permits seamless integration between the website login and the opening of PDFs.
- Can be machine independent.
- Supports "online only" functionality, where user may open files only while logged-in
- May be selectively bypassed via ErrorMode=SendAnyway

Con:

- Requires that users accept permanent cookies.
- Imposes browser dependency. The current client retrieves cookies from the following browsers:
 - o Windows: IE, Netscape, Mozilla, Firefox
 - o Macintosh: IE, Netscape, Mozilla, Safari, Camino and Firefox

Username/Password Authentication:

The Username/Password (Authent=p) permits the PermissionServer to present a customizable dialog to the user and request credentials. The UNP strings are not stored in the document, but rather are both defined and validated by the PermissionServer, so may be issued and withdrawn at will.

Pro:

- Usable either Online or Offline.
- Can be machine independent.
- No browser dependency, or required; can be used online or from CD.
- Frequency of dialog appearance is customizable (default is once per session, but may be changed to once per document, once per hour, etc.)
- May be selectively bypassed via ErrorMode=SendAnyway

Con:

- If used online will force a duplicate login (once for the website, once for the PDF)
- Requires distribution of UNP data via external channel

Machine Authentication:

MachineID Authentication (Authent=m) is the default method, at least insofar as the machine identifiers are sent with the above methods. To be effective, this approach requires a two-stage procedure in which the user first registers a machine (via a dynamically encrypted file delivered while that user is logged-in, for example) and subsequent requests are processed by validating one or more of the machine context identifiers.

Pro:

- Usable either Online or Offline.
- No browser dependency, or required; can be used online or from CD.
- Transparent; no user interaction required.

Con:

- Typically requires additional machine-registration step.
- Does not distinguish between users on the same machine⁷⁹

Machine Identifiers

The client and protocol support three main device identification mechanisms, these are:

DiskID is built from the identifier of the first disk (the volume number written on the disk when it is hard formatted) and CPU data. It will be the same for all users of the machine.

MachineID: is built using a part of the MAC address of the first card having a MAC address. It will be the same for all users of the machine, but may change if the network connection type is changed (e.g. the machine goes from a wired connection to a wireless one, or vice versa).

UUID: is a Universally Unique Identifier. It is created the first time the plug-in loads then is memorized by the plug-in. This will be different for each user. It should never change, but could in principle be deleted by the user and in that case would be re-generated. If this happens, the client notifies the server of this fact, via the UUIDisNew=true message.

The plug-in also sends the identity of the logged-in user, from the OS login as the parameter User=. Note however that while versions of the client up to 0802 deliver this value by default, from now on the server must request that this information be sent by adding the text SendUserLogin=true in the response to the settings request.

The PermissionServer may therefore manage a matrix of four identifiers, of these two are most useful: DiskID will identify the machine and UUID will identify a login on that machine.

⁷⁹ Note that the User= value may be parsed to distinguish users, though this is not necessarily unique and may be generic ("Administrator").

Errors Defined by PDFEncryptor2

Error Condition	Error Number
ERROR_NONE	0
ERROR_CANNOT_OPEN_INPUT	1
ERROR_CANNOT_OPEN_OUTPUT	2
ERROR_ADD_NUMBER_NO_MEMORY	3
ERROR_NO_LENGTH_IN_STREAM_OBJECT	4
ERROR_WRITE_ERROR	5
ERROR_READ_ERROR	6
ERROR_BAD_STRING_LENGTH_ATTRIBUTE	7
ERROR_NUMBER_REF_NOT_FOUND	8
ERROR_NO_STREAM_BUFFER_MEMORY	9
ERROR_ENCRYPTION_FAIL	10
ERROR_STREAM_OBJECT_REF_NOT_FOUND	11
ERROR_NO_STARTXREF_FOUND	12
ERROR_FSEEK_FAILED	13
ERROR_FPUTS_FPRINTF_FAIL	14
ERROR_UNEXPECTED_READ_ERROR_OR_EOF	15
ERROR_NO_SIZE_IN_EXISTING_TRAILER	16
ERROR_FILE_ALREADY_ENCRYPTED	17
ERROR_INPUT_AND_OUTPUT_DIRS_ARE_SAME	18
ERROR_NO_MEMORY_FOR_OBJECTS	19
ERROR_LONG_NOT_SAME_SIZE_AS_INT	20
ERROR_INVALID_OBJECT_NUMBER	21
ERROR_LINE_ENDED_IN_STRING	22
ERROR_UNEXPECTED_BACKSLASH_ERROR	23
ERROR_STRING_TOO_LONG	24
ERROR_DUFF_HEX_STRING	25
ERROR_NO_EOF_FOUND	26
ERROR_BAD_ROOT_IN_TRAILER	27
ERROR_NO_ROOT_IN_EXISTING_TRAILER	28
ERROR_CANNOT_OPEN_KEYVALUE_FILE	29
ERROR_NO_EQUALS_SIGN_IN_KEYVALUE_FILE_LINE	30
ERROR_NO_NAME_OR_VALUE_IN_KEYVALUE_FILE_LINE	31
ERROR_ROOT_OBJECT_NOT_FOUND	32
ERROR_LONG_LONG_NOT_64	33
ERROR_ENCRYPTED_STRING_TOO_SMALL	34

Errors Defined by PDFEncryptor3

Encryptor library

Class	Error #	Error name		
Encryptor	9051	MemoryAlloc		
	9052	InputFileNotFound		
	9053	OutputFileNotFound		
	9054	LicenseError		
	9055	NotEncryptLicense		
	9056	NotValid		
	9057	InvalidLicenseSet		
	9058			
	9059	InvalidInfoSettings		
	9060	DocumentError		
	9061	UnmanagedError		
	9065	DeLinearized		
CryptInfo				
XchInfo				

A typical error will be in the form:

- 9060 == Document error (the encryptor finds an error in the document)
- 1623 == Ill formed PDF file == The document was first created as a linearized document (it has a linearized dic) but was modified later (it has multiple xref tables).

There are two ways to encrypt such a document:

- The safe one: open it then resave it linearized (perhaps with another name) before encrypting it.
- The unsafe one: set the force bit; there is no guarantee that the encrypted file will be well-formed.

A complete list of PDFEncryptor errors is available upon request.

Acrobat Language Codes

Language	LCID	ISO4Char	RFC1766
Chinese—Simplified	CHS	zh-cn	zh-cn
Chinese—Traditional	CHT	zh-tw	zh-tw
Danish	DAN	da-dk	da
Dutch	NLD	nl-nl	nl
English	ENU	en-us	en
Finnish	SUO	fi-fi	fi
French	FRA	fr-fr	fr
German	DEU	de-de	de
Italian	ITA	it-it	it
Japanese	JPN	ja-jp	ja
Norwegian	NOR	no-no	no
Portuguese—Brazil	PTB	pt-br	pt-br
Spanish	ESP	es-sp	es
Swedish	SVE	sv-se	sv

Document Perms

None	Open	Print	Edit	Copy	EditNotes	SaveAs	Total Val	PermType
0								pdPermNone
	1						1	pdPermOpen
		4					4	pdPermPrint
			8				8	pdPermEdit
				16			16	pdPermCopy
					32		32	pdPermEditNotes
						64	64	pdPermSaveAs
	1	4					5	pdPermOpenPrint
	1		8				9	pdPermOpenEdit
	1	4	8				13	pdPermOpenPrintEdit
	1			16			17	pdPermOpenCopy
	1	4		16			21	pdPermOpenPrintCopy
	1		8	16			25	pdPermOpenEditCopy
	1	4	8	16			29	pdPermOpenPrintEditCopy
	1				32		33	pdPermOpenEditNotes
	1	4			32		37	pdPermOpenPrintEditNotes
	1		8		32		41	pdPermOpenEditEditNotes
	1	4	8		32		45	pdPermOpenPrintEditEditNotes
	1			16	32		49	pdPermOpenCopyEditNotes
	1	4		16	32		53	pdPermOpenPrintCopyEditNotes
	1		8	16	32		57	pdPermOpenEditCopyEditNotes
	1	4	8	16	32		61	pdPermOpenPrintEditCopyEditNotes
	1					64	64	pdPermOpenSaveAs
	1	4				64	64	pdPermOpenPrintSaveAs
	1		8			64	64	pdPermOpenEditSaveAs
	1	4	8			64	64	pdPermOpenPrintEditSaveAs
	1			16		64	64	pdPermOpenCopySaveAs
	1	4		16		64	64	pdPermOpenPrintCopySaveAs
	1		8	16		64	64	pdPermOpenEditCopySaveAs
	1	4	8	16		64	64	pdPermOpenPrintEditCopySaveAs
	1				32	64	64	pdPermOpenEditNotesSaveAs
	1	4			32	64	64	pdPermOpenPrintEditNotesSaveAs
	1		8		32	64	64	pdPermOpenEditEditNotesSaveAs
	1	4	8		32	64	64	pdPermOpenPrintEditEditNotesSaveAs
	1			16	32	64	64	pdPermOpenCopyEditNotesSaveAs
	1	4		16	32	64	64	pdPermOpenPrintCopyEditNotesSaveAs
	1		8	16	32	64	64	pdPermOpenEditCopyEditNotesSaveAs
	1	4	8	16	32	64	64	pdPermOpenPrintEditCopyEditNotesSaveAs

A

Absolute Expiration, 57

C

CD-ROM, 5
 commandline, 10
 Communications Protocol, 5, 7, 18, 64
 configuration data, 7

D

Decryption, 7
 DocPerm request, 32
 DocPerm values, 39
DocumentID, 8, 54, 55, 57, 64, 65

E

encryption, 8
 Encryption, 7
Encryption Key, 66

F

FileOpen client, 5, 7
 FileOpen Client, 14
 freeBSD, 9

H

http, 7
 https, 7

L

license file, 10
 Linux, 9

M

machineID, 8
 metadata, 5, 7
 MSWindows, 9

O

Offline Access, 5
 Offline permission, 53
 Offline Permissions file, 5
Online Access, 5
Online/Offline Hybrid Access, 5

P

Parameter File, 10
 PDFEncryptor, 10
 Perl, 18
permissionFile, 54, 56
permissionPDF, 55
 PrintPerm request, 48

R

Relative Expiration, 57

S

ServiceID, 8, 54, 56, 65
 session cookie, 8
 Sun Solaris, 9

T

Trace File, 16

U

username/password, 8